INTERFACE AGE

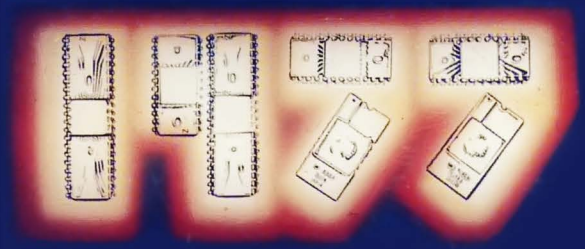
MICROCOMPUTING FOR HOME AND SMALL BUSINESS VOL. 2, ISSUE 2 JAN. 1977 $1.50

AMI'S EVK Series Microcomputer Boards

Building a 3 Digit A/D Converter

CONSOL 1K Resident Operating System

Livermore 8080 BASIC



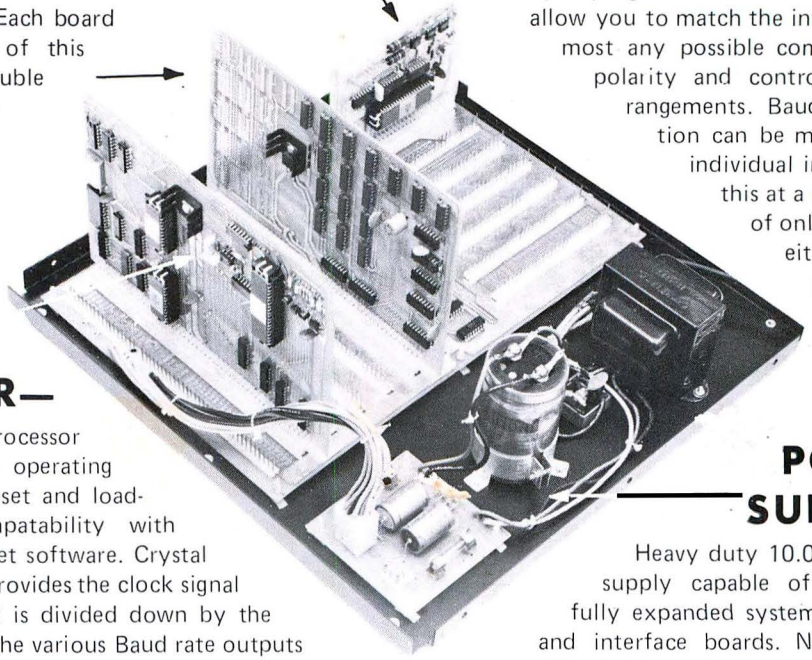
WHY SETTLE FOR LESS—
THAN A 6800 SYSTEM

MEMORY—

All static memory with selected 2102 IC's allows processor to run at its maximum speed at all times. No refresh system is needed and no time is lost in memory refresh cycles. Each board holds 4,096 words of this proven reliable and trouble free memory. Cost—only $125.00 for each full 4K memory.

INTERFACE—

Serial control interface connects to any RS-232, or 20 Ma. TTY control terminal. Connectors provided for expansion of up to eight interfaces. Unique programmable interface circuits allow you to match the interface to almost any possible combination of polarity and control signal arrangements. Baud rate selection can be made on each individual interface. All this at a sensible cost of only $35.00 for either serial, or parallel type



PROCESSOR—

"Motorola" M6800 processor with Mikbug® ROM operating system. Automatic reset and loading, plus full compatability with Motorola evaluation set software. Crystal controlled oscillator provides the clock signal for the processor and is divided down by the MC14411 to provide the various Baud rate outputs for the interface circuits. Full buffering on all data and address busses insures "glitch" free operation with full expansion of memory and interfaces.

POWER SUPPLY—

Heavy duty 10.0 Amp power supply capable of powering a fully expanded system of memory and interface boards. Note 25 Amp rectifier bridge and 91,000 mfd computer grade filter capacitor.

DOCUMENTATION—

Probably the most extensive and complete set of data available for any microprocessor system is supplied with our 6800 computer. This includes the Motorola programming manual, our own very complete assembly instructions, plus a notebook full of information that we have compiled on the system hardware and programming. This includes diagnostic programs, sample programs and even a Tic Tac Toe listing.

Mikbug® is a registered trademark of Motorola Inc.

SwTPC 6800
Computer System

with serial interface and 4,096 words of memory. $395.00

☐ Enclosed is $395 for my SwTPC Computer Kit ☐ Send Data

☐ or BAC_____ #_____

☐ or MC_____ Ex Date_____

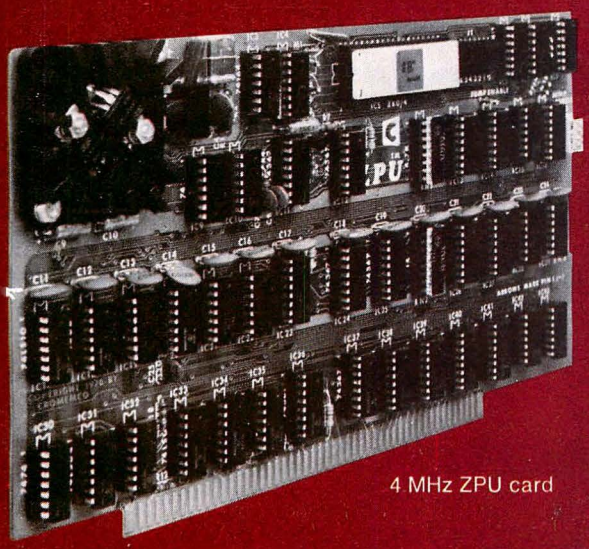
NAME_____

ADDRESS_____

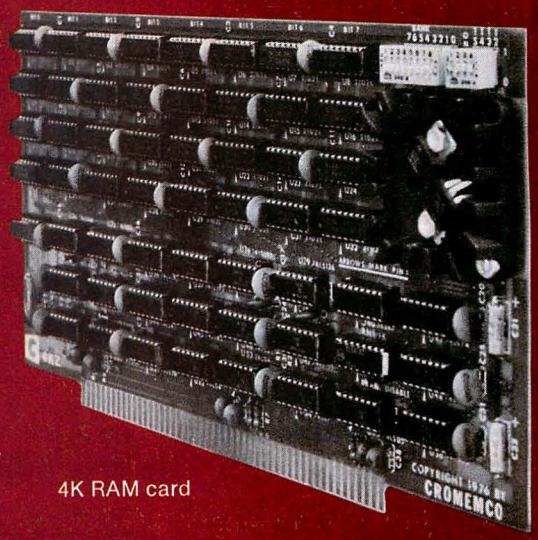
CITY_____ STATE_____ ZIP_____

Southwest Technical Products Corp., Box 32040, San Antonio, Texas 78284

CIRCLE NO. 31 ON INQUIRY CARD



4 MHz ZPU card



4K RAM card

How your best choice of CPU cards can have a really-expandable memory

LIKE TO A HALF MEGABYTE

You probably know our Z-80 CPU card. It's the most powerful card available. Not only does it have a guaranteed speed of 4 MHz and a crystal-controlled 2/4 MHz clock rate, **it also has a power-on memory jump feature** that greatly simplifies starting-up.

Now we've developed an outstanding 4K RAM memory card for this CPU card (or for any S-100 bus CPU card). **Our new Model 4KZ is a static memory that has:**

(1) a guaranteed speed of 4 MHz
(2) a memory-bank-select feature.

As you would expect with a Cromemco product, this new Model 4KZ gives you advanced performance at low cost. It achieves its 4 MHz speed while using proven, reliable low-power memory chips (21L02's). How? By design that uses address anticipation.

ENORMOUSLY EXPANDABLE

You get staggering expandability in the new 4KZ — to 512 kilobytes if you'd like.

Here's how: with the 4KZ you can organize memory into as many as 8 banks of 64K bytes each.

Then an 8-position switch on the 4KZ selects a given bank.

With memory expandability like that, Cromemco's CPU and RAM cards are the basic hardware for a broad range of jobs — even jobs that until now were only for large computers.

LOW PRICED

The new 4KZ has the high quality Cromemco is known for. It is available at computer stores — or directly from the factory.

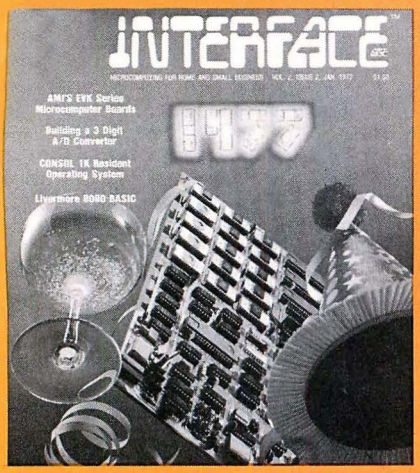
Just a word of caution. The 4KZ is bound to be in demand, so we suggest you act promptly.

4K Static RAM Memory kit (Model 4KZ-K) . . . $195
4K Static RAM Memory assembled, burned-in and tested (Model 4KZ-W) $295

ZPU card kit (Model ZPU-K) $295
ZPU card assembled, burned-in and tested (Model ZPU-W) $395

Cromemco
incorporated
Specialists in computers and peripherals
2432 CHARLESTON RD., MOUNTAIN VIEW, CA 94043 • (415) 964-7400

CIRCLE INQUIRY NO. 8



INTERFACE AGE™

MICROCOMPUTING FOR HOME AND SMALL BUSINESS

Cover Story

Almost daily the microcomputer industry celebrates new advances as more and more creative and ingenious ideas are turned into state-of-the-art products.

Because of the increasing versatility enabled by the wide breadth of low-cost peripherals, today's user can now approach a fairly sophisticated microcomputer system and maintain financial solvency. This month's feature, "AMI's EVK Series Microcomputer" is the story of just one such microcomputer whereby a standard prototype board becomes the common vehicle for four configurations based on the user's memory and application needs.

DEPARTMENTS

HARDWARE FEATURES

SOFTWARE FEATURES

INTERFACE AGE Magazine, published monthly by McPheters, Wolfe & Jones, 13913 Artesia Blvd., Cerritos, Calif. 90701. Subscription rates: U.S. $10.00, Canada/Mexico $12.00, all other countries $18.00. Opinions expressed in by-lined articles do not necessarily reflect the opinion of this magazine or the publisher. Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes an endorsement of the product or products by this magazine or the publisher.

POSTMASTER: Please send change of address form 3579 and undelivered copies to **INTERFACE AGE** Magazine, 13913 Artesia Blvd., Cerritos, Calif. 90701. Second-class postage paid at Artesia, California 90701 and at additional mailing offices.

Turn us on.

And the Digital Group will get you going.

One of the first things you'll discover when you get started with microprocessors is that there's a lot more involved than the hardware. That's why you should consider a system's software, too.

With a Digital group system, you can get going fast...and you don't have to be a programming genius to make your computer **do** something for you. Turn us on, and your system really does what you want it to do—easily and quickly. Because the Digital Group firmly believes that a computer without usable software is useless.

Every system the Digital Group delivers has several operating programs included with it. As soon as you turn it on it's doing something! In addition, we make available an ever-growing number of software packages for Digital Group systems at all levels of support. (They're listed below.)

But first, how do you get started?

With Tiny Basic Extended.

For only $5, you get the "beginning" and for an additional $5 you can get a cassette full of game programs that work with it. Both are on audio cassette that your Digital Group system can understand. You can list the programs on the TV screen of your Digital Group system and see exactly how they work step by step.

Now the real fun begins.

Change the program. See what happens. Make it work better. Try other variations. The best way to learn how to program your computer is by studying an easy-to-understand operating program and changing it to see what happens. Bit by bit, understanding will come. (And it's nice to know that in the meantime, your system can be working while you're learning.)

When you're ready for more, so are we.

Here are a few programs we have for you:

TINY BASIC EXTENDED **$5.00**

TINY BASIC GAMES:

- Volume 1—Chomp, Checkers, Tic-tac-toe, Digiguess and Brainteaser **$5.00**
- Volume 2—Artillery, Dr. Therapy, Reverse, Biorythm and Golf **$5.00**
- Volume 3—Taxman, Snark, Trap, Number, Square-Root and Clock **$5.00**
- Volume 4—Hamurabi, Stars, 23-Matches, 20-Questions, Blackjack, Factor and Batum .. **$5.00**
- Volume 5—Bomber, Lunar Lander, Spacebattle, Matador and Dice **$5.00**
- Volume 6—Chief, Mr. Quizzer, Addition, Subtraction and Multiplication **$5.00**

"GALAXY"
1976, SCELBI Computer Consulting, Inc. ... **$7.50**

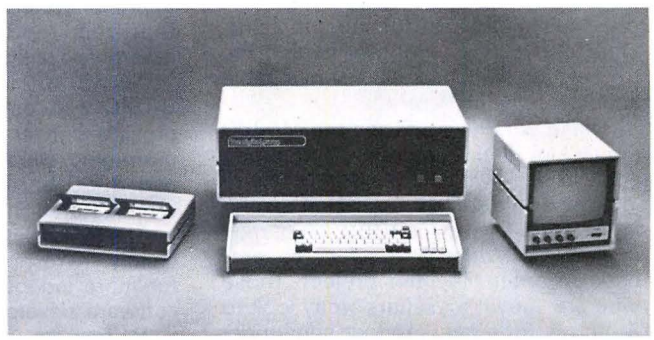
ALSO:

- Z-80 Educator **$10.00**
- Z-80 Assembler **$15.00**
- Z-80 Dis-Assembler **$10.00**
- Z-80 Text/Editor **$7.50**

And many more.

Here's how to get going.

First, take a look at our hardware (we've pictured it here in our new line of cabinets). Then just fill in the coupon below for all the details on our systems—hardware and software—so you can turn us on.



the digital group INC

P.O. Box 6528
Denver, Colorado 80206
(303) 777-7133

OK, Get me going. I want all the details.

Name _____

Address _____

City/State/Zip _____

INTERFACIAL

The beginning of the new year finds both consumer and manufacturer with high expectations for a very dynamic market place. Semiconductor manufacturers are readying to launch their new microcomputers into the arena such as Intel's 8085 (single supply, micro system on a chip), Fairchild's new microcomputer system on a card and Motorola's new microcomputer HEP kit.

The message seems to be coming through loud and clear as most of the new entries from the manufacturers are aimed at the low cost market place with improved performance over their more costly ancestors.

To start your microcomputing year off with a sizzling success we have compiled a bonanza of indepth hardware and software entries which we expect you will find as an invaluable addition to your library.

Those who have utilized the M6800 family in their personal computing applications, will find the first article of interest as a detailed discussion involving a CRT Controller application may inspire the "do-it-yourselfer" to embark on another winter project. Those who are "buy it myselfers" may look to Motorola for a new product announcement. Nevertheless Joe Roy and Dusty Morris have offered a good insight into what it takes.

Part II of a four part series on AMI's Microcomputers cover the EVK Prototype card architecture and a comparative analysis of each of the four basic configurations offered.

"Building a 3 digit A/D converter" can not only be a challenging job but can be fun if most of the toil is removed by virtue of a single chip. Roger Edelson explores the Siliconix LD130 bi polar A/D chip and applications in a microcomputer with a two part series starting this month.

Terry Benson, Field Applications Engineer for Intel Corp, turns his prolific pen loose on "Microcomputer Design Aides." A bit of a squint is provided for another new tool, the PROMPT 80™ a self contained microcomputer development system which supports the design of any 8080 system. The PROMPT 80™ is reputedly a personal programming tool as well as a professional's programming delight.

The "Card-of-the-Month" feature is a mind soother and program expeditor. Around for awhile and still in great demand, the Cromemco BYTESAVER™. is reviewed with a lusty zeal as our hardware editor pokes and prys at its features and tests its performance.

A resident 8080 software package, "CONSOL" developed by Processor Technology for their new Intelligent Microcomputer Terminal called SOL is featured including the complete assembly listing.

Two BASIC interpreters, one on Tiny BASIC and one on standard BASIC provide conversational language programming capabilities for the SC/MP and 8080 based Microcomputers are also featured in this issue.

The features in this month's issue are chiefly in response to many requests for more indepth product reviews and software development programs.

Remember we try to be responsive to your needs and the only way we can succeed is through your continued communication. Feel free to drop us a line and tell us what you would like to see in future issues.

If you have composed an article on a subject you feel others would be interested in, let us take a look at it. If we concur, we will publish it and that also means it could make you some money — as well as satisfy that creative pen.

INTERFACE AGE will pay up to $50 per published page as well as award each monthly "best article" author an additional $100.00. The annual "best article" award is $500 in cash or equipment.

Editor

INTERFACE AGE
MAGAZINE

13913 Artesia Boulevard
Cerritos, CA 90701
(213) 926-6629

PUBLISHER & EDITOR-IN-CHIEF
ROBERT S. JONES

ASSISTANT EDITOR
LINDA FOLKARD-STENGEL

GENERAL MANAGER
NANCY A. JONES

ASSISTANT MANAGER
EVA YAKA

NORTHWESTERN REGIONAL EDITOR
ADAM OSBORNE, PhD

NEW ENGLAND REGIONAL EDITOR
ROBERT E. TRIPP

EDITORIAL CO-ORDINATOR
WILLIAM SEVEDGE

HARDWARE EDITOR
ROGER EDELSON

SOFTWARE EDITOR
ROBERT STEVENS

ASSOCIATE EDITOR
JANE HILL ALDEN

PRODUCTION LAYOUT
ROB STUART

ART
CHARLYNE LEVANDOWSKI

SUBSCRIPTION CIRCULATION
JO ANN FERGUSON

NATIONAL SALES MANAGER
BRUCE BERKEY

ADVERTISING—Representative
STEVE RICHARDS
ZACH BOVINETTE

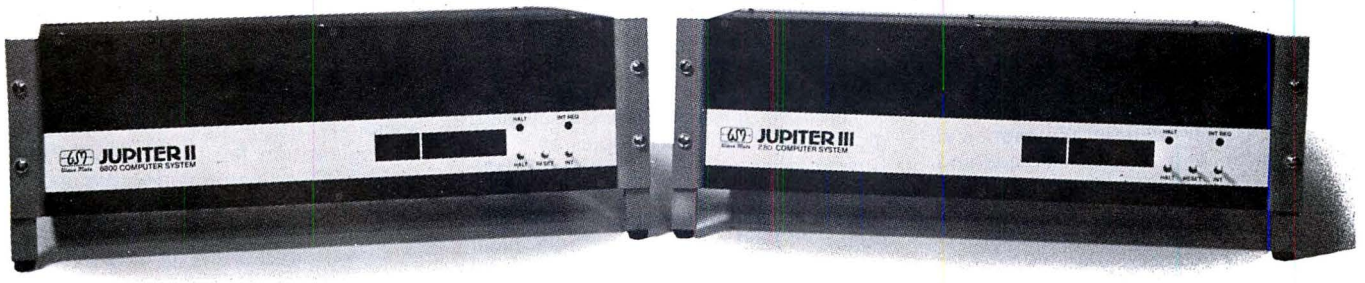
RETAIL CIRCULATION
ZACH BOVINETTE

Editorial Correspondence
Direct all correspondence to the appropriate editor at: INTERFACE AGE magazine, P.O. Box 1234, Cerritos, CA 90701. Editorial contributions must be accompanied by return postage and will be handled with reasonable care, however, publisher assumes no responsibility for return or safety of manuscripts, art work, or models.

Advertising Inquiries
Direct all advertising inquiries to: Advertising Department, INTERFACE AGE magazine, 61 South Lake Avenue, P.O. Box 4566, Pasadena, CA 91106. (213) 795-7002.

THE BEGINNING

OF A BEAUTIFUL FRIENDSHIP



JUPITER II A
6800 System
$795

JUPITER III A
Z80 System
$865

If you thought the quality of a wire-wrapped system was beyond your price range — Take a look at what we have now!

The Jupiter IIA and the Jupiter IIIA Basic computer systems. You get the system module cage with fully assembled backplane, fully assembled plug-in ferro-resonant power supply, front panel kit and your choice of 6800 or Z80 CPU module kit. All less than the price of the two best selling 8080 systems!

Plus you can choose from the fastest growing selection of memories and peripherals available from any manufacturer, like our 2KB EPROM/ 4KB RAM/serial RS-232 module which can transform your basic computer system into a real tiger.

And remember, all Wave Mate products meet the highest quality industrial standards, with rugged construction unmatched by anyone! So don't let any grass grow under your feet, get yours now. Write or call for more info and your closest Wave Mate authorized distributor.

You get your choice of microprocessors! And you get wire-wrapped modules too!

Now you have a low cost way to get started into personal computing without sacrificing future growth capability!

☐ Send details on Jupiter II systems
☐ Have salesman call
Name _____
Title _____
Company _____
Address _____
City _____ State _____ Zip _____
Phone _____

Wave Mate

WAVE MATE 1015 West 190th Street, Gardena, California 90248 Telephone (213) 329-8941
Dept. 20

CIRCLE INQUIRY NO. 35

UPdate

NOTABLES

"Who are we?" "How did we get here?" "Are we alone in the Universe?" The answers to these questions have broad scientific implications, as well as deep philosophical, sociological, and theological meaning. The search for extraterrestrial intelligence has begun!

The Forum for the Advancement of *Students in* Science and Technology (FASST), the Student Programs Division of the American Institute of Aeronatuics and Astronautics (AIAA), and the Educational Programs Office of the NASA/Ames Research Center, invite you to take part in the search by participating in a special symposium, "The Search for Extraterrestrial Intelligence (SETI)." The program will be held at the Ames Research Center, near San Francisco, on February 24-25, 1977.

Speakers and participant discussion groups will cover such topics as: "The Cosmic Picture — Is Anyone Really Out There?"; "The Origin of Life — Chemical and Biological Considerations"; "Evolution of Technological Civilizations"; "Methods and Technology for the Search"; and "Cultural Implications of Detection and Contact with Extraterrestrial Intelligence."

Confirmed speakers for the symposium include Dr. Richard Berendzen, astronomer, lecturer, author and Chairman for the Boston SETI symposium, "Life in the Mind of Man"; Dr. John Billingham, Chief, Program Office for SETI-NASA/Ames; Ronald Bracewell, Professor/Stanford Univeristy and author of THE GALACTIC CLUB; David Black, Project scientist/SETI; Dr. James Christian, philosopher and author of EXTRATERRESTRIAL INTELLIGENCE: The First Encounter; and Bernard Oliver, Vice President for Research, Hewlett-Packard, and formerly Director of the study group on constructing large arrays of radio telescopes, designated Project Cyclops. We are awaiting confirmation of Dr. Carl Sagan, astronomer/biologist, and Dr. Stanley Miller, biochemist.

Directed, primarily, toward college and university students, the symposium also welcomes participation by interested professionals and faculty members. A wide cross section of academic disciplines will be involved, including students in anthropology, theology, life science, engineering, sociology, etc.

Registration fee for the conference is $10.00, which includes symposium materials, a tour of the Ames facilities, and a banquet. The grant to conduct this program did not include funds for student travel. For those who would like to participate but live outside the San Francisco area, we recommend that you seek financial assistance locally from your academic department or campus student activity fund. In your request for funding, include a statement on how the conference applies to your educational objectives, and the benefits to be derived from attending. FASST will send support letters upon your request. Requests may be addressed to FASST, 1785 Massachusetts Avenue, N.W., Washington, D.C. 20036. Telephone (202) 483-2900.

The BYTE SHOP has recently opened a second store in Phoenix, at 12864 N. 28th Drive (Westown). Like the first store opened last October at 813 N. Scottsdale Road, BYTE SHOP — WEST features microcomputers for use in the home, business or school at prices comparable to stereo systems.

The phenomenon of personal computing is new and growing. Both owners, Alan P. Hald and Jeffrey D. McKeever, were recently interviewed on a TV newspot distributed nationwide. Jeff remarked, "Suddenly we received calls from all over the country, from businessmen, lawyers, housewives, engineers and students intrigued by the idea of personal computing." Both felt that the response confirmed recent articles in Newsweek, Time and Business Week which indicated that many people throughout the country are becoming computer enthusiasts. Alan noted, "In Phoenix, the response has been encouraging. Often we find ourselves staying open after closing hours engrossed with our customers

G R-R-R-R e a t

The NEW Sphere DOS – a Real Tiger!!

When we decided to redesign our already successful Disk Operating System, we asked ourselves how we would want it done. Based on a year of solid experience building, shipping and supporting a good DOS, we decided we would want our new DOS to be able to:

- provide a complete Editor usually found only with minis
- offer a full-blown Assembler (not just a small mnemonic set)
- handle variable-length files
- recognize and handle both stream and block-oriented files
- deal with multiple devices of several types
- operate on both our 300 and 500 systems.

So, we designed the all-new Sphere DOS, and we have a lot of good reasons for calling it a tiger! It is a truly remarkable DOS . . . worth checking into. Once you do, we think you'll find it easy to make the decision to

PUT THIS TIGER ON YOUR TEAM!!!

DOS SPECIFICATIONS

LANGUAGE	Motorola M6800 Assembler
STORAGE REQUIREMENT	5K Bytes
OPERATING ENVIRONMENT	Sphere 300 and 500 Series Systems
CONVENIENCE PROGRAMS PROVIDED	Editor, Debugger, Assembler, Set of File-manipulating Commands (macro-like operations)
PERIPHERALS SUPPORTED	CRT, Keyboard, Floppy Disk Drive, TTY
STRUCTURE	Uniform, interrupt-driven I/O

Personal and commercial computer product distributorships available. For information call 801-292-8466.

CIRCLE INQUIRY NO. 30

SPHERE CORPORATION

940 North 400 East
North Salt Lake, Utah 84054 • *(801) 292-8466*

developing new games and applications."

Both stores conduct introductory classes and provide free in-store lectures for schools and civic groups. The owners feel that increasing public awareness of the potential for personal computing is a major part of their business. They both have been active in speaking to local organizations about the future and microcomputing. Both owners are looking forward to bringing the fun and capability of personal computing to Arizonans.

For further information contact either Alan or Jeffry at (602) 894-1129 or (602) 942-7300.

CALL FOR PAPERS

The UPIEEE-77 Workshop (June 1977) "On Beach Programming of Microprocessors" is soliciting papers from people who have incorporated microprocessors into a product without the use of expensive MP program development equipment. Three types of papers are expected — software, hardware and trouble-shooting (debugging). All hardware papers accompanied by a working board will be published in the Proceedings.

In the 1950s the physicists gave us the transistor. Before we could conveniently use it in circuit design, we had to strip it of the theory with which the physicists had saddled it, and provide it with a practical engineering approach. Now the computer people have given us the microprocessor. Before we can conveniently use it, we have to strip it of the top-heavy computer approach and provide it with a practical engineering approach for use in circuit design.

The purpose of the μPIEEE-77 workshop is to exchange experiences and learn pitfalls and shortcuts in writing firmware; and in trading-off firmware, software and hardware right at the design bench. Papers are due before February 1.

The workshop will take place on Friday, Saturday and Sunday, June 10, 11 and 12, in Philadelphia. Dormitory space will be available to out-of-town participants. Proceedings will be published beforehand and mailed to participants. There will be no reading of papers — only discussion, questions and answers, and demonstration of boards. Boards chosen for publication only will be displayed in the lobby, and authors will be given time for poster talks. Time will be allocated for bull sessions.

For further information about μPIEEE-77 contact Miss Helen B. Yonan, Philadelphia IEEE, Moore School, University of Pennsylvania, Philadelphia, PA 19174.

TENTH ONE-DAY TECHNICAL SYMPOSIUM

The Los Angeles Chapter of the Association for Computing Machinery has announced final arrangements for its Tenth One-Day Tehnical Symposium.

Date: February 4, 1977.

Location: Hoffman Hall, University of Southern California, Los Angeles.

Theme: Computers . . . the Continuing Revolution.

Registration Fee: Early - $15.00, Late - $20.00. Fee covers programs and lunch.

Morning Program: Software, the new emphasis.

Speakers and their subjects will be Dr. Robert Brown, Hughes Aircraft Company, "Principle Aspects of Structured Design"; Dr. Jack Holton, University of Southern California, "Modern Techniques for Programming System Development — Are They Useful?";Guy de Balbine, Caine Farber Gordon, Inc., "An Assessment of Structured Processor Technology: Present and Future"; Prof. Brinch Hansen, Univeristy of Southern California, "The Underlying Architecture of the Concurrent PASCAL Machine."

The lunch will be a complete steak dinner. Luncheon speaker will be Victor Azgapetian, Aerojet General, "A Look Into the Future From the Wrong Point of View."

Now you can buy an Altair™ 8800b or an Altair 680b computer right off the shelf. Altair plug-in boards, peripherals, software and manuals are also available. Check the list below for the MITS dealer in your area.

off the shelf.

RETAIL COMPUTER STORE, INC.
Tim & Susanne Broom
410 NE 72nd St.
SEATTLE, WA 98115
(206) 524-4101

COMPUTER KITS (S. F. area)
Pete Roberts
1044 University Ave.
BERKELEY, CA 94710
(415) 845-5300

THE COMPUTER STORE
(Arrowhead Computer Co.)
Dick Heiser
820 Broadway
SANTA MONICA, CA 90401
(213) 451-0713

GATEWAY ELECTRONICS, INC.
George Mensik OF COLORADO
2839 W. 44th Ave.
DENVER, CO 80211
(303) 458-5444

COMPUTER SHACK
Pete Conner
3120 San Mateo NE
ALBUQUERQUE, NM 87110
(505) 883-8282, 883-8283

GLOBAL ENGINEERING CO.
5416 South Yale
TULSA, OKLA. 74145
(918) 452-2567

COMPUTER PRODUCTS UNLIMITED
Harry & Margaret Mohrmann
4216 West 12th
LITTLE ROCK, AR 72204
(501) 666-2839

GATEWAY ELECTRONICS, INC.
Harry & Margaret Mohrmann
Lou Elkins, Stuart Bartfield
8123-25 Page Blvd.
ST. LOUIS, MO 63130
(314) 427-6116

CHICAGO COMPUTER STORE
Lou Van Eperen
517 Talcott Rd.
PARK RIDGE, IL 60068
(312) 823-2388

THE COMPUTER ROOM
3938 Beau D'Rue Drive
Eagan, MN 55122
Dale Hagert, Bob Raemer
(612) 452-2567

BYTE'TRONICS
John & Stan Morrow
Suite 103
1600 Hayes St.
NASHVILLE, TN 37203
(615) 329-1979

THE COMPUTER SYSTEMCENTER
Jim Dunion, Rich Stafford,
Steven Mann, Ron Roberts
3330 Piedmont Road
ATLANTA, GA 30305
(404) 231-1691

THE COMPUTER STORE, INC.
Sid Halligan
120 Cambridge St.
BURLINGTON, MA 01803
(617) 272-8770
Jeff Feldman, Service Dept.

THE COMPUTER STORE OF NEW YORK
Bob Arning
55 West 39th St.
NEW YORK, NEW YORK 10018
(212) 221-1404

THE COMPUTER STORE OF
Peter Blond ANN ARBOR
310 East Washington Street
ANN ARBOR, MI 48104
(313) 995-7616

THE COMPUTER STORE, INC. (Hartford area)
George & Susan Gilpatrick
63 South Main Street
WINDSOR LOCKS, CT 06096
(203) 627-0188

MICROSYSTEMS (Washington, D.C.)
Gloria & Russell Banks
6605A Backlick Rd.
SPRINGFIELD, VA 22150
(703) 569-1110

THE COMPUTER STORE
Stephen Payne
1114 Charleston National Plaza
CHARLESTON, W. VA. 25301
(304) 343-4607

MARSH DATA SYSTEMS
Don Marsh
5405 B Southern Comfort Blvd.
TAMPA, FL 33614
(813) 886-9890

mits

CIRCLE INQUIRY NO. 19

Afternoon Program: Hardware, the micro revolution. Speaking will be Ron Muir, ROM Systems, "Emulation Utilizing the 2900 Microprocessor Bit Slice Chip Set"; Elizabeth Rather, Forth, Inc., "FORTH as a Programming Support System for Microprocessor"; and Les Lazar, Digital Design and Development, "Cost Versus Performance Tradeoffs for Several Microprocessors."

PANHANDLE COMPUTER SOCIETY AMARILLO, TEXAS

The Texas Panhandle now has a computer hobbiest club and though less than 3 months old, we boast a membership of more than 30.

The Panhandle Computer Society meets every Friday and is open to anyone interested in computers. We also welcome replies from other groups, especially in Texas.

Those interested may contact the president, Tex Everett, 2923 S. Spring, Amarillo, Texas 79103 (Phone: 806-373-8207) or the secretary/treasurer: Jerry Fewell, 3109 Browning, Amarillo, Texas 79103 (phone 806-374-0897.

CALENDAR

Jan. 3 South Florida Computer Group, Miami Division meets at 7:30 p.m., PAP Institute, 1155 NW 14th St., Miami, FL. General Meeting. For more information call (305) 271-2805.

Jan. 4 El Paso Computer Group meets at 7:00 p.m.. Meeting place information call (915) 544-1542.

Jan. 5 Northwest Computer Club meets at 7:00 p.m., Pacific Science Center, Seattle, WA. Informal Meeting. Details by calling (206) 524-6359.

Jan. 5 New England Computer Society, Inc. meets at 7:00 p.m., MITRE Corp. cafeteria, Route 62, Bedford, MA. For further information call Dave Day (603) 434-4239.

Jan. 5 Southern California Computer Society Valley Chapter meets at 7:00 p.m., Harvard School, 3700 Cold Water Canyon, Studio City, CA. Technical session on the TI-9900 Microprocessor by a representative of Texas Instruments Co. Call John Scott at (213) 849-4094 for further details.

Jan. 8 The Permian Basin Computer Group, Midland Chapter meets at 4:00 p.m., Midland College, Occupational Technology Bldg., Room 110. More details call (915) 697-4607.

Jan. 8 The Permian Basin Computer Group, Odessa Chapter meets at 1:00 p.m., Odessa College, Electronics Technology Bldg., room 203. Please call (915) 332-9151 for further details.

Jan. 8 Louisville Area Computer Club meets at 1:00 p.m., Speed Auditorium, University of Louisville. General meeting. Contact Glen Darwin (502) 456-5589 or by writing at 3028 Hunsinger Ln., Louisville, KY 40200.

Jan. 9 Southern California Computer Society, Orange County Chapter meets at 12:00 noon, California State University, Fullerton, Administration Bldg., Room 321. For meeting agenda call: Lorin Mohler at (714) 998-5831.

Jan. 9 University of Chicago Computer Club meets at 1:00 p.m., Reynolds Club, 57th and University, Chicago, IL. For more information call (312) 421-0244.

Jan. 9 Oklahoma Computer Club meets at 10:00 a.m., Bell Aisle Library, Oklahoma City, OK. General Meeting. Call Al Campbell (405) 842-4933.

Jan. 13 Rochester Area Microcomputer Society meets at 6:30 p.m., Room 1030, Bldg. 9, Rochester Institute of Technology, Rochester, NY. Meeting will feature William Mathis and the inside scoop on the KIM-1. Mailing address, RAMS, P.O. Box D, Rochester, NY 14609.

Jan. 14 Cresent City Computer Club meets at 8:00 p.m., University of New Orleans, Lakefront Campus. Contact Bob Latham (504) 722-6321 or P.O. Box 1097, New Orleans, LA 70122.

Jan. 15 South Central Kansas Amateur Computer Association meets at 9:00 a.m., 1430 E. Kellog (next to Church of the Nazarene). Call Cris Borger at (316) 945-9658 for more information.

Jan. 15 So. Cal. SWTPCO MP 6800 Users Group meets at A-VID Electronics, 1655 E. 28th Street, Long Beach, CA 90806 at 10:00 a.m.

Jan. 19 Denver Amateur Computer Society DACS meets at 7:30 p.m., 4101 E. Hampden Av. Denver, CO. General Meeting. Call (303) 333-1047.

Apple Introduces the First Low Cost Microcomputer System with Video Terminal and 8K Bytes of RAM on a Single PC Card.

The Apple Computer. A truly complete microcomputer system on a single PC board. Based on the MOS Technology 6502 microprocessor, the Apple also has a built-in video terminal and sockets for 8K bytes of on-board RAM memory. With the addition of a keyboard and video monitor, you'll have an extremely powerful computer system that can be used for anything from developing programs to playing games or running BASIC.

Combining the computer, video terminal and dynamic memory on a single board has resulted in a large reduction in chip count, which means more reliability and lowered cost. Since the Apple comes fully assembled, tested & burned-in and has a complete power supply on-board, initial set-up is essentially "hassle free" and you can be running within minutes. At $666.66 (including 4K bytes RAM!) it opens many new possibilities for users and systems manufacturers.

You Don't Need an Expensive Teletype.

Using the built-in video terminal and keyboard interface, you avoid all the expense, noise and maintenance associated with a teletype. And the Apple video terminal is six times faster than a teletype, which means more throughput and less waiting. The Apple connects directly to a video monitor (or home TV with an inexpensive RF modulator) and displays 960 easy to read characters in 24 rows of 40 characters per line with automatic scrolling. The video display section contains its own 1K bytes of memory, so all the RAM memory is available for user programs. And the Keyboard Interface lets you use almost any ASCII-encoded keyboard.

The Apple Computer makes it possible for many people with limited budgets to step up to a video terminal as an I/O device for their computer.

No More Switches, No More Lights.

Compared to switches and LED's, a video terminal can display vast amounts of information simultaneously. The Apple video terminal can display the contents of 192 memory locations at once on the screen. And the firmware in PROMS enables you to enter, display and debug programs (all in hex) from the keyboard, rendering a front panel unnecessary. The firmware also allows your programs to print characters on the display, and since you'll be looking at letters and numbers instead of just LED's, the door is open to all kinds of alphanumeric software (i.e., Games and BASIC).

8K Bytes RAM in 16 Chips!

The Apple Computer uses the new 16-pin 4K dynamic memory chips. They are faster and take ¼ the space and power of even the low power 2102's (the memory chip that everyone else uses). That means 8K bytes in sixteen chips. It also means no more 28 amp power supplies.

The system is fully expandable to 65K via an edge connector which carries both the address and data busses, power supplies and all timing signals. All dynamic memory refreshing for both on and off-board memory is done automatically. Also, the Apple Computer can be upgraded to use the 16K chips when they become available. That's 32K bytes on-board RAM in 16 IC's—the equivalent of 256 2102's!

A Little Cassette Board That Works!

Unlike many other cassette boards on the marketplace, ours works every time. It plugs directly into the upright connector on the main board and stands only 2" tall. And since it is very fast (1500 bits per second), you can read or write 4K bytes in about 20 seconds. All timing is done in software, which results in crystal-controlled accuracy and uniformity from unit to unit.

Unlike some other cassette interfaces which require an expensive tape recorder, the Apple Cassette Interface works reliably with almost any audio-grade cassette recorder.

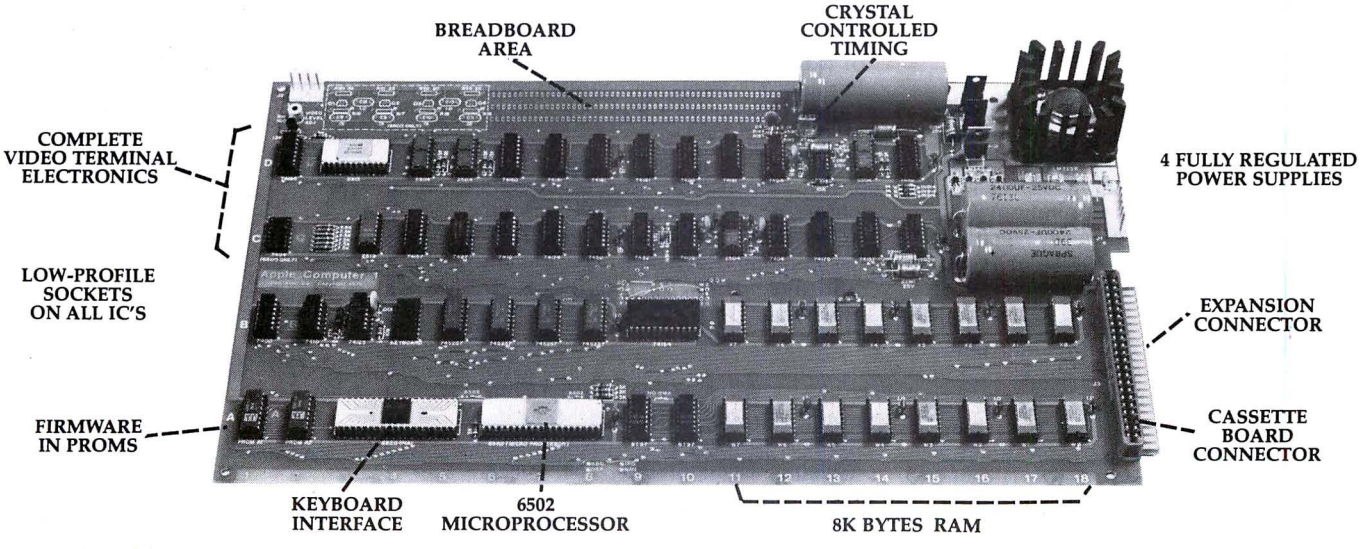
Software:

A tape of **APPLE BASIC** is included free with the Cassette Interface. Apple Basic features immediate error messages and fast execution, and lets you program in a higher level language immediately and without added cost. Also available **now** are a dis-assembler and many games, with many software packages, (including a macro assembler) in the works. And since our philosophy is to provide software for our machines free or at minimal cost, you won't be continually paying for access to this growing software library.

The Apple Computer is in stock at almost all major computer stores. (If your local computer store doesn't carry our products, encourage them or write us direct). **Dealer inquiries invited.**

Byte into an Apple $666.66*

* includes 4K bytes RAM



APPLE Computer Company • 770 Welch Rd., Palo Alto, CA 94304 • (415) 326-4248

Jan. 19 Northwest Computer Club meets ar 7:00 p.m., Pacific Science Center, Seattle, WA. General Meeting. Details of agenda: call (206) 524-6359.

Jan. 20 South Florida Computer Group, Ft. Lauderdale Division meets at 7:30 p.m., FPL, 501 S. Andrews, Ft. Lauderdale, FL. Call (305) 522-5358 for more details.

Jan. 22 Minnesota Computer Society, 7:30 p.m., Hennepin County Library, Southdale Branch, 7001 York Ave South, Edina, MN. For meeting agenda write, MCS, P.O. Box 35317, Minneapolis, MN 55435.

Jan. 30 Chicago Area Computer Hobbyists Exchange (CACHE) meets at 12:00 noon, NIGAS Bldg., Shermer Rd., Glenview IL. LSI-11 Conference, Digital Equipment Corporation Show. Call Bill Precht at (312) 620-1671 or P.O. Box 36, Vernon Hills, IL 60061.

Feb. 1 El Paso Computer Group meets ar 7:00 p.m. Meeting place by calling (915) 544-1542.

Feb. 2 New England Computer Society, Inc. meets at 7:00 p.m., MITRE Corp., Cafeteria, Route 62, Bedford MA. Call Dave Day for more information at (603) 434-4239.

Feb. 5 Louisville Area Computer Club meets at 1:00 p.m., Speed Auditorium, University of Louisville. General meeting. Contact Glen Darwin (502) 456-5589 or by writing at 3028 Hunsinger Ln., Louisville, KY 40200.

Feb. 7 AMRAD Amateur Radio Research and Development Corp. meets at 8:00 p.m. in the Patrick Henry Library, Vienna, VA 22101. Call (703) 356-8918 for details.

Feb. 11 Crescent City Computer Club meets at 8:00 p.m., University of New Orleans, Lakefront Campus. For more information contact Bob Latham (504) 722-6321.

Feb. 11 Rochester Area Microcomputer Society, 6:30 p.m., Room 1030, Bldg. 9, Rochester Institute of Technology, Rochester, NY. Mailing address, RAMS, P.O. Box D, Rochester, NY 14609.

Feb. 12 The Permian Basin Computer Group, Midland Chapter meets at 4:00 p.m., Midland College, Occupational Technology Bldg., Room 110. General Meeting. Call (915) 697-4607 for details.

Feb. 12 The Permian Basin Computer Group, Odessa Chapter meets at 1:00 p.m., Odessa College Electronics Technology Bldg., Room 203. Call (915) 332-9151 for more information.

Feb. 12 So. Cal. SWTPO MP 6800 Users Group meets at 10:00 a.m. at A-VID Electronics, 1655 E. 28th Street, Long Beach CA 70806.

Feb. 27 Chicago Area Computer Hobbyists Exchange (CACHE) meets at 12:00 noon, NIGAS Bldg., Schermer Rd., Glenview IL. Sick Computer Show. Bring in your problems and we'll help repair them. Call Bill Precht at (312) 620-1671.

Feb. 28-Mar. 3 IEEE Computer Society's COMPCON '77 Spring beginning at 9:00 a.m. on Feb. 28 at the Jack Tar Hotel, San Francisco, California.

Mar. 2 New England Computer Society, Inc., meets at 7:00 p.m., MITRE Corp., Cafeteria, Route 62, Bedford, MA. General Meeting. Call Dave Day at (603) 434-4239.

Mar. 7 AMRAD Amateur Radio Research and Development Corp. meets at 8:00 p.m. in the Patrick Henry Library, Vienna, VA 22101. Call (703) 356-8918 for details.

Mar. 10 Rochester Area Microcomputer Society, 6:30, Room 1030, Bldg. 9, Rochester Institute of Technology, Rochester, NY. Mailing address, RAMS, P.O. Box D, Rochester, NY 14609

Mar. 27 Chicago Area Computer Hobbyists Exchange (CACHE) meets at 12:00 noon, NIGAS Bldg., Schermer Rd., Glenview, IL. Small Business Opportunities Show. For further information contact Bill Precht (312) 620-1671.

This Space Awaits

YOUR Cartoon

and

Your Bank Awaits

Our CHEQUE

send in your humor

And now minis too! All from a catalog, at discount prices.

Digital's Direct Sales Catalog — the first catalog to offer computer products by mail with off-the-shelf delivery — was such a success, we've come out with an expanded second edition that includes the PDP-8A, the newest member of the world's most popular minicomputer family.

We've also added the LA180 line printer and expanded the sections on microcomputers, logic modules, terminals, cabinets, connectors, accessories, and supplies from the first edition.

Of course, you still get our 5% catalog discount, plus another 4% for cash with your order. You also get a 10-day free trial period, plus our standard 90-warranty. (Since we're selling by mail, you do have to install the equipment yourself.)

If you were among the thousands who ordered our original catalog, you'll get the new one automatically by mail.

If you weren't, why not sign up now?

For your free copy, call 800-225-9480 (Mass. 617-481-7400 ext. 6608). Or write: Components Group, Digital Equipment Corp., MR2-2/M59, One Iron Way, Marlborough, MA 01752. (Catalog sales to U.S. only.)

CIRCLE INQUIRY NO. 9

Letters to the Editor

Dear Editor:

I would like to thank you for having taken the time to discuss with me my problem in finding any MIKBUG documentation.

I would like to comment on the above problem. I am an engineer at a small company which provides analysis and programming services as well as designing test apparatus and providing test conductors. This work has been done both for larger companies and directly for government agencies, primarily branches of the Navy. My own areas are dynamics and system modeling. My programming is done in FORTRAN and my method of computer access is thru TSO. I have become aware of the possibilities of the microcomputer and the various available peripherals for my own use in such areas as text editing and for use by our test personnel in various ways. I have accordingly set out to try to educate myself in the ways of the microcomputer both for the good of my company and for selfish reasons. I think that engineers far removed from electronics will need to understand the intelligent machines to continue to be fully useful. At any rate, the point I was going to make is that my personal quest for information will shape the views I pass on to my employer(s), and if I find myself in a position to recommend some device or some supplier, it will be one who has been both willing and capable of supplying me with information, as well as being interested in providing access to their product in small or unit lots. This isn't to suggest that they supply free samples, but rather that they see to it that there are retail dealers who are knowledgable as to their merchandise and who are responsive to small orders. It is a case of casting their bread upon the water . . . I would suspect that a significant percentage of the people who are interested in microprocessors, etc. as an avocation or for home applications are professionals in fields in which these same devices might find use. Thus, a manufacturer who gives adequate attention to the personal computing market is probably effectively advertising in the commercial market which he sees as his real target. I would like to suggest that you carry on a dialogue with the major manufacturers along these lines and base an article on it.

Thank you for the interesting and useful magazine which you produce and thank you in advance for the MIKBUG data.

Philip Sollow

Dear Editor:

I presently own two microcomputers, one is a 6800 based system and the other is an 8080 based system, and I am searching for software with which to operate my computers for my business.

I am particularly interested in programs that involve inventory control, payroll, ledgers, journals, balance sheets, income statements, profit and loss statements, invoicing and in general any programs which will perform basic bookkeeping tasks and accounting tasks for a small business.

I would very much appreciate any information that you have concerning software that is business related, as well as any other categories that you might have available (such as games etc.). Such information should include a description of the program, your price, and the length of the program. Any speed and consideration that you could lend to this matter will be greatly appreciated. Thanking you, I remain

James E. Preston

There is a three volume set of books called BASIC Software Library published by R. W. Brown of SRI and available from the Microcomputer Software Depository that includes small business programs — see the Book Review section of this issue for more information.

Software Editor

Dear Editor:

I am a home computer hobbyist presently utilizing a homebrew 8080 system. I have a cassette interface and a TV typewriter. I am looking to expand and/or update my system with the latest hardware developments. Also of interest is software especially BASIC.

Please send me any information you have concerning your software products. Your cooperation is appreciated.

Rupert Fenequito

Dear Editor:

I am enclosing a SASE and a check for $10, payable to Interface Age, for a one year subscription.

Please sent me your list of available software for 8080A based microcomputers. My interests are primarily two fold: (1) Scientific programming (for which I will need FORTRAN or Extended BASIC compilers) and (2) Data Processing for which I could use a good monitor and DOS.

Dennis Eisen

Dear Editor:

I just received my copy of the November issue of INTERFACE AGE and would like to congradulate you on the job you're doing in the Software area. I hope that we can get a Cassette Operating System for the 8080 as a contributed program in the near future.

In your introduction on page 92, you indicated in the second paragraph that an 8080 program by T. W. Travis on memory diagnostics would be published. It appears that it is missing. It is also referenced in the Software Depository Listing on page 128 as appearing in VI NIZ. I trust it will be published in the next issue.

You also indicated that voting for the best article of the month would have to be accomplished by the last date of the month on the issue. As you can see that may eliminate those who receive the magazine late. How about the middle of the following month?

By the way, is there anyone in the SCCS who responds to letters? I know that INTERFACE AGE is not connected with SCCS but from the East Coast it appears that SCCS is dead.

It would be helpful to make a point on the software of indicating how much memory is required for the program and how much memory is required to run the program — when possible — and also if the program can be in PROM or whether RAM is necessary. This would definitely be of help in the Depository listings.

Thanks again for instituting a means of obtaining low cost software.

R. I. Demrow

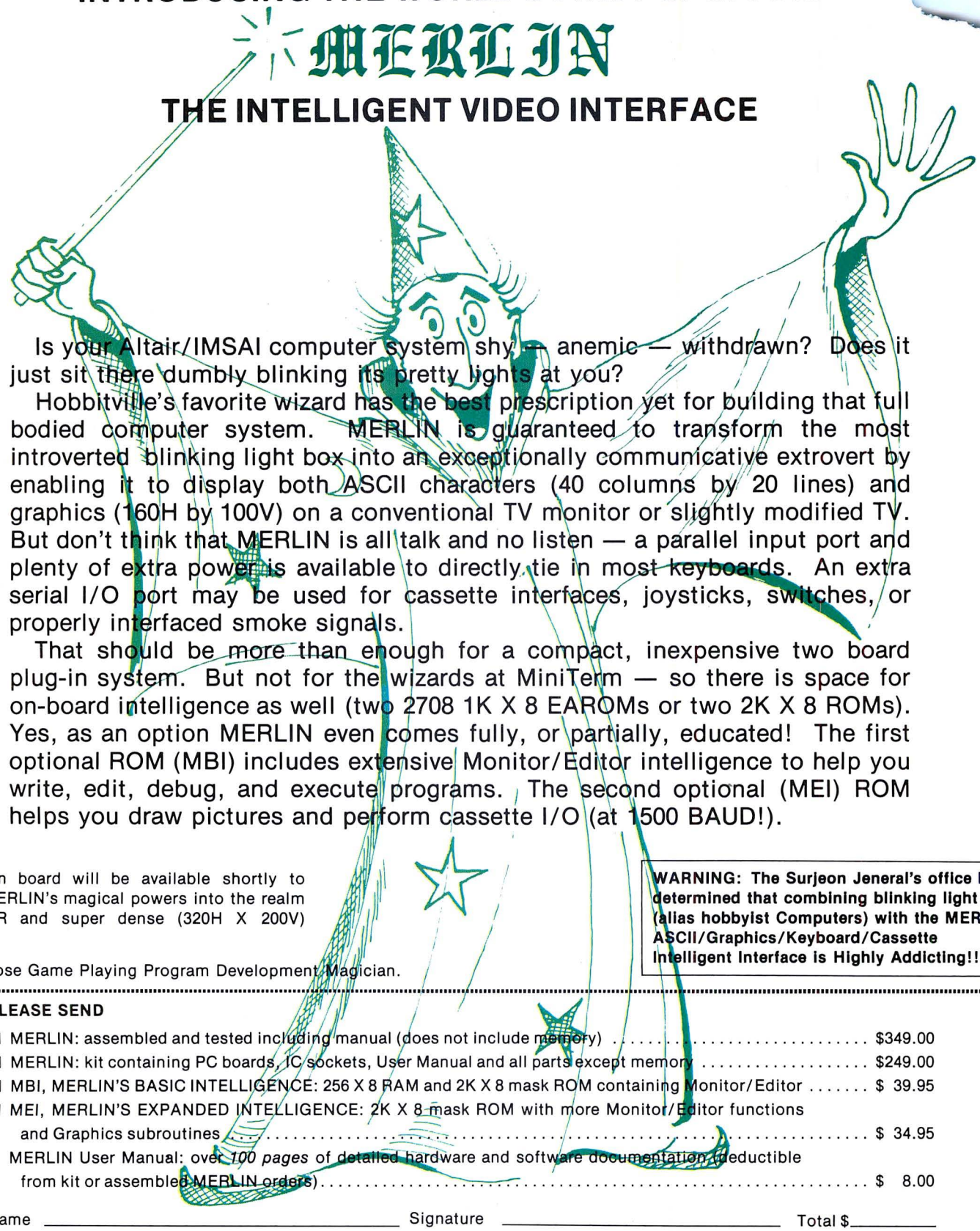
As you now know we ran out of space for the November issue and had to publish the article Memory Diagnostics by T. W. Travis in the December issue of INTERFACE AGE. We goofed and left all references to this article in the November issue.

*Best article of the month voting deadline is the month **following** date of publication.*

INTRODUCING THE WORLD'S FIRST *GPGPPDM**

MERLIN

THE INTELLIGENT VIDEO INTERFACE



Is your Altair/IMSAI computer system shy — anemic — withdrawn? Does it just sit there dumbly blinking its pretty lights at you?

Hobbitville's favorite wizard has the best prescription yet for building that full bodied computer system. MERLIN is guaranteed to transform the most introverted blinking light box into an exceptionally communicative extrovert by enabling it to display both ASCII characters (40 columns by 20 lines) and graphics (160H by 100V) on a conventional TV monitor or slightly modified TV. But don't think that MERLIN is all talk and no listen — a parallel input port and plenty of extra power is available to directly tie in most keyboards. An extra serial I/O port may be used for cassette interfaces, joysticks, switches, or properly interfaced smoke signals.

That should be more than enough for a compact, inexpensive two board plug-in system. But not for the wizards at MiniTerm — so there is space for on-board intelligence as well (two 2708 1K X 8 EAROMs or two 2K X 8 ROMs). Yes, as an option MERLIN even comes fully, or partially, educated! The first optional ROM (MBI) includes extensive Monitor/Editor intelligence to help you write, edit, debug, and execute programs. The second optional (MEI) ROM helps you draw pictures and perform cassette I/O (at 1500 BAUD!).

P.S. An add-on board will be available shortly to expand MERLIN's magical powers into the realm of COLOR and super dense (320H X 200V) graphics!

*General Purpose Game Playing Program Development Magician.

WARNING: The Surjeon Jeneral's office has determined that combining blinking light boxes (alias hobbyist Computers) with the MERLIN ASCII/Graphics/Keyboard/Cassette Intelligent Interface is Highly Addicting!!!

PLEASE SEND

- ☐ MERLIN: assembled and tested including manual (does not include memory) $349.00
- ☐ MERLIN: kit containing PC boards, IC sockets, User Manual and all parts except memory $249.00
- ☐ MBI, MERLIN'S BASIC INTELLIGENCE: 256 X 8 RAM and 2K X 8 mask ROM containing Monitor/Editor ..... $ 39.95
- ☐ MEI, MERLIN'S EXPANDED INTELLIGENCE: 2K X 8 mask ROM with more Monitor/Editor functions and Graphics subroutines ..... $ 34.95
- ☐ MERLIN User Manual: over *100 pages* of detailed hardware and software documentation (deductible from kit or assembled MERLIN orders) ..... $ 8.00

Name _____ Signature _____ Total $_____

Street _____ City _____ State _____ Zip _____

- ☐ More information
- ☐ Check Enclosed
- ☐ Master Charge # _____ Exp. Date_____
- ☐ BankAmericard # _____ Exp. Date_____

MiniTerm Associates, inc.

Box 268, Bedford, Mass. 01730 (617) 648-1200

CIRCLE INQUIRY NO. 18

A CRT Terminal Using the M6800 Family

By Joe Roy and Dusty Morris
Data Processing Systems Engineering
Motorola Inc., Phoenix, AZ

This article describes a versatile M6800 based CRT Controller for "glass-teletype," smart, programmable, and intelligent CRT terminals. While a complete duplication of the entire package may be beyond the capabilities of most readers, some of the design features should be of particular utility in other construction projects. Of particular interest is the exploitation of the biphase clock architecture of the M6800 system, providing higher throughput, more I/O handling capability, less interference patterns during refresh memory accesses, and easier task-orientated multiple processor implementation in a CRT terminal than is possible with other approaches. Let's look at the features of this CRT terminal:

1. 24 rows of 80 characters.
2. 7 X 9 uppercase characters in a 9 X 12 dot block; shifted lower case through the use of a custom programmed MCM6832 16K Binary ROM.
3. Conventional non-interlaced raster scan.
4. Blink, half-intensity, video invert, underline, and non-display FACS (Field Attribute Codes); embedded FACS with optional widened memory capability.
5. Alternating inverted/non-inverted cursor.
6. 50/60 Hz field rate is logic selectable; display is centered for both 50 and 60 Hz.
7. Transparent accesses of Refresh Memory by VIA and MPU.
8. Limited graphics implementation with no changes in basic design philosophy.
9. Design philosophy facilitates up-grading a simple economical terminal with upward compatible software and hardware to a task-oriented multiple processor intelligent terminal.
10. MPU is unburdened from overhead of refresh memory contention and is free to service keyboard, edit functions, serial communications, and high speed control such as floppy disk.

The basic configuration for a microprocessor-controlled CRT terminal is shown in Figure 1. An MC6800 microprocessor executes the CRT terminal executive firmware routine and jumps to driver subroutines when servicing the keyboard, serial synchronous or asynchronous interface, floppy disk formatter, and other peripherals. Cursor movements, R/W, and all editing functions are programmable and under microprocessor control. Actual refresh of the CRT monitor display is done with a configuration of SSI/MSI hardware called a VIA (Video Interface Adapter). The VIA provides video, vertical sync, and horizontal sync to the Motorola M3000 (or equivalent) monitor. The monitor must meet the requirements specified in Figure 2.

The MPU and VIA share the CRT Refresh Memory. Since the processor clock is derived from the VIA, both are synchronized. As shown in Figure 1 timing diagram, the VIA accesses memory for CRT refresh during clock phase Ø1, while the MPU accesses memory during Ø2. The Refresh Memory is organized in an odd address block and an even address block. Both an even and the adjacent odd address characters are transferred during a Ø1 access, whereas a single character is transferred to the MPU during a Ø2 access. The "odd/even memory" concept allows characters to be pulled from memory at a rate ($\simeq$ 2 MHz) sufficient to update the CRT. The "interleaved clocking" of memory makes it look transparent to both the MPU and the VIA. That is, neither delays the access of the other to memory. Consequently, less MPU overhead results than in other approaches.

Note that the interleaved clocking of memory is unaffected by cycle stretching of either MPU Ø1 or Ø2 . . . techniques used for refreshing dynamic memories, synchronizing other I/O, or interfacing slow memory.

The Refresh Memory provides a bidirectional data bus to the MPU and a two byte output bus for screen refresh. The two bytes of display data are pipelined to even and odd latches which are alternately enabled as data to the address inputs of an MCM6832 character ROM. The address is an ASCII character which the

•Single M6800 Microprocessor Controls CRT, Keyboard, Serial Communications Interface, and Jloppy Disk in Smart/Intellegent CRT terminal.

•Unique 2-Port Refresh Memory Configuration eliminates contention overhead for memory accesses.

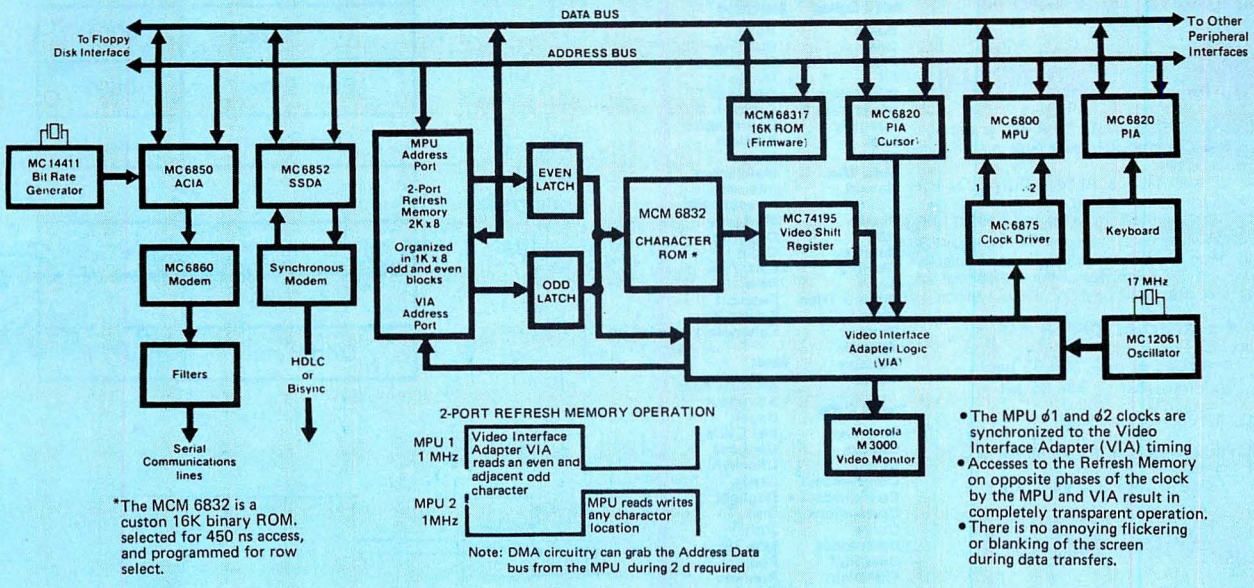
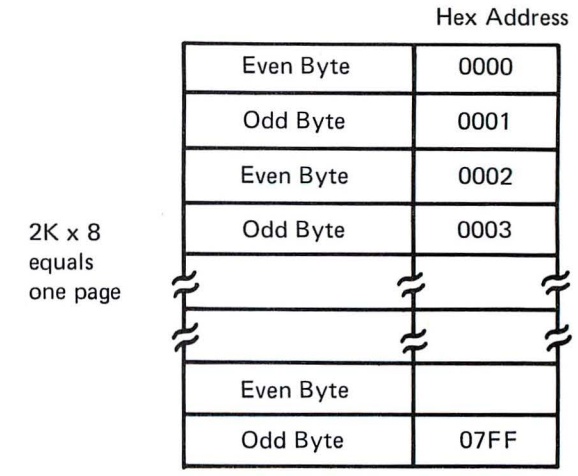


Figure 1. M6800 Terminal Block Diagram

1.	Character Matrix	(Columns	7	7	
2.		(Rows	9	9	
3.	Character Block	(Columns	9	9	
4.		(Rows	12	12	
5.	Frame (refresh) Rate		50 HZ	60 HZ	
6.	Rows of Characters		24	24	
7.	Active Scan Lines (line 6 times line 4)		288	288	
8.	Delay before Vertical Sync (No. of scan lines)		(1667ms) 31	0	
9.	Vertical Sync Width (No. of scan lines)	4.52 ms	(215ms) 4	4 (215ms)	1.18 ms
10.	Delay after Vertival Sync (No. of scan lines)		(2634ms) 49	18 (968ms)	
11.	Total Scan Lines (Line 7+8+9+10)		372	310	
12.	Horizontal Frequency (line rate) (line 11 times line 5)		(53.76ms) 18.6 KHZ	18.6 KHZ (53.76ms)	
13.	Character Rate (line 12 times line 18)		1.8972 MHZ	1.8972 MHZ	
14.	Characters/Rows		80	80	
15.	Delay before Horiz. Sync. μsec:	(Character times)	(2.1ms) 4	4 (2.1ms)	
16.	Horizontal Sync Width μsec:	(Character times)	(4.7ms) 9	9 (4.7ms)	
17.	Delay after Horiz. Sync. μsec:	(Character times)	(4.7ms) 9	9 (4.7ms)	
18.	Total character times (line 14+15+16+17)		102 (11.5ms)	102 (11.5ms)	
19.	Clock Rate (line 13 times line 3)		17.074800 MHZ	17.074800 MHZ	
20.	Display size		12″	12″	
21.	Character Time (reciprocal of line 13)		527 ms	527 ms	
22.	MPU Clock (line 19÷2X line 3)		948.6 KHZ	948.6 KHZ	
23.	Clock Time (reciprocal of line 19)		58.6 ms	58.6 ms	

Figure 2 Video Monitor Timing (Motorola Display Products M3000 Monitor meets these requirements)



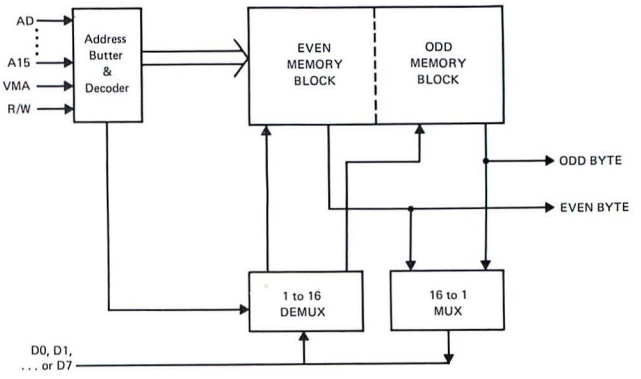


Figure 3.

ROM maps into a block of dots. The particular row of dots (0 to 11) in the block is determined by four "row select" inputs from the VIA. In a raster scan system, each ASCII character is presented 12 times to the character ROM with a sequential row select each time in order to paint the complete character on the screen.

Each row of dots is parallel loaded into an 8 bit shift register and clocked out serially. Field Attribute Codes (FACS) such as inverted video are imposed on the serial data stream by the VIA before the video is sent to the monitor.

This section describes the 2 port memory technique which allows interleaving of the MPU and display functions with minimal interference. Figure 3 is a simplified block diagram of the functional implimentation. As can be seen the memory is divided into two blocks; one 'even' and one 'odd.' Selection of the appropriate block is by address line AO, while A1 through A10 select one of the 1024 bytes in each block. A11 through A13 are used to select the particular "page" of memory. R/W and VMA are used in the read/write process and to determine if data is gated 'in' or 'out' on the data line DO through D7.

As described earlier, the refresh RAM is organized as a 1K X 8 even block and a 1K X 8 odd block. Even and odd blocks are interleaved to form a 2K X 8 "page." Of the 2048 bytes, 1920 (80 times 24) are required per page of display, leaving 128 bytes spare. Because the refresh memory looks like any other RAM on the MPU bus, this spare 128 bytes are free for scratch and the stack. In multiple page systems, it serves as an edit buffer.

The refresh memory is implemented with 450ns 2102 style 1K X 1 memories. Even and odd blocks each contain eight devices. The new 2114 style 1K X 4 static memories (spec'd at 450ns) are an attractive alternate (only four are required per page). Memory addressing is through 1 of 2 ports. Referring to Figure 4, the schematic offers the two port memory, we will look into the detailed design.

The VIA address counter (DA1-DA10) is gated with a set of MC6887 high speed three state buffers, the Motorola equivalent of the 8797 device. The enable signal to pins 1 and 15 on U20 and U21 is generated only during EN DISP ADDR at P2 pin 3 (roughly 01 interval) and when PAGE SELECT is high at P2 pin 4. The latter signal is only required in multiple page systems, and determined which page is displayed. Note that pin 15 on U20 is always enabled. This gates pin 12 to 11 path buffers VMA (Valid Memory Address). This signal is for gating the MPU address buffers (U18 and U19). When used with systems such as the EXORcisor where a block of addresses must be unconditionally protected, this signal should be VUA (Valid Users Address).

The first complete small computer under $1,000

Twenty-five years ago a computer so powerful cost you a cool million.

Today, for less than 0.1% of the cost, you can have your own personal computer with even greater power—one that fits in a package no larger than a typewriter.

It's the Sol-20 Terminal Computer. There's nothing like it anywhere.

A powerful computer at your disposal is quite simply, fantastic!

Use your computer to run your ham station. Do your taxes and books with it. Control your inventory. Curb your energy bill by letting your computer regulate energy usage. Use it to study science and engineering. Use it to help you invent.

And after hours, take your computer home to play (and create) sophisticated games. Computers may indeed be the ultimate hobby because you never outgrow them. Uses for computer intelligence are literally unlimited.

You can find a basic computer kit for about $600. Though by the time you purchase the other components needed to make it run—keyboard, additional memory, software and I/O interfaces—you're up around $1,500. Phew!

Now, the good news. The new Sol-20 in kit form gives you an entire working system for $995.

Those of you who are already into personal computers will recognize what an incredible advance this is in computer packaging. This is the only small computer that offers all of the following as *standard* features:

8080 microprocessor—1024 character video display circuitry—control PROM memory—1024 words of static low-power RAM—1024 words of preprogrammed interface capable of controlling two recorders at 1200 baud—both parallel and serial standardized interface connectors—a complete power supply including fan—a beautiful case with solid walnut sides—software which includes a preprogrammed PROM personality module and a cassette with BASIC-5 language plus two sophisticated computer video games—the ability to work with all S-100 bus, including Altair, Imsai and Processor Technology products.

Full expansion capability

Tailor the Sol-20 system to your applications with our complete line of peripheral products. These include the video monitor, audio cassette and digital tape systems, dual floppy disc system, memories, and interfaces.

Write for our new 22 page catalog. Get all the details.

Processor Technology, Box G, 6200 Hollis St., Emeryville, CA 94608. (415) 652-8080.

Processor Technology Corporation

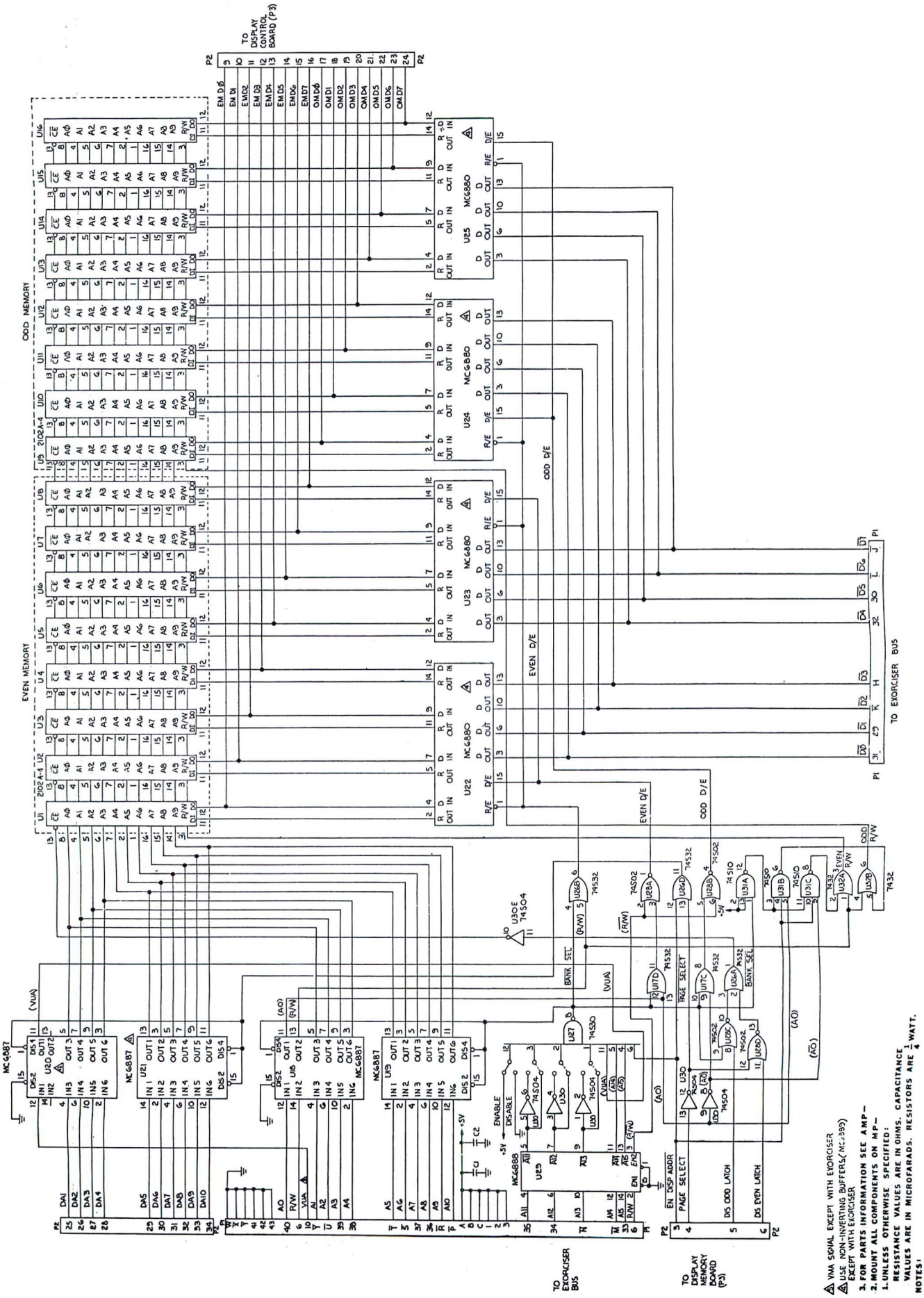


Figure 4. Display Memory Board

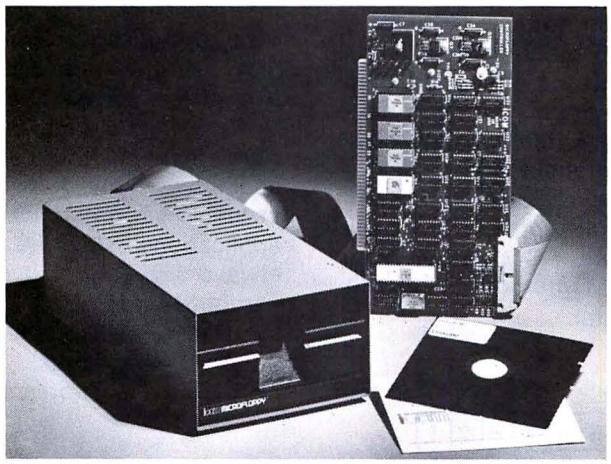
Guess who just bought our new Microfloppy™ for $1095 including software?

Plug-compatible for $1095

iCOM's brand new Microfloppy™ is specifically designed for the personal computer user. It's smaller, using the new 5¼-inch diskette, and is priced at only $1095 including disk drive, power supply, cabinet, controller/interface card, all cables and connectors, plus, on diskette, iCOM's new FDOS-M software, including assembler, editor, and file manager. And for a limited time,* you also get an 8K Disk BASIC package at no extra cost.

Best of all, iCOM's Microfloppy™ is fully assembled and tested. It's 100% compatible with Altair, IMSAI, Poly 88 and other microcomputers using the Altair bus format.

*Offer good through April 15, 1977



Much faster than a cassette

This compact, economical Floppy Disk System is 10 to 15 times faster than a cassette. Programs can be loaded in seconds, a source program of several thousand characters can be corrected and a new file stored on diskette in less than a minute. Any of up to 175 files can be reached in 1.5 seconds.

iCOM — the leader

We've been building Floppy Disk Systems for microcomputers for nearly three years. Thousands of our systems are operating perfectly in the field. Many major computer manufacturers have incorporated our Floppy Disks into their systems. We deliver, and we'll be around to give you service whenever you may need it. You see, we're a division of Pertec Computer Corp., one of the largest and most respected manufacturers of peripherals, microsystems, data entry products, and data processing systems.

So you don't need to risk getting burned by a fly-by-night company. Order an iCOM® Microfloppy™ and be sure of getting the best.

See iCOM's Microfloppy™ now at your local computer store. Or contact us for the name of your dealer.

icom® MICROPERIPHERALS®

a division of Pertec Computer Corporation

6741 Variel Ave., Canoga Park, CA 91303 U.S.A.
(213) 348-1391 TWX 910-494-2788

The selection of U18 and U19 is decoded by U27 which generates BANK Select. The particular bank (1 of 8 pages) is strap selectable (U29 and U30). The ENABLE/DISABLE switch on U27 provides a means of overlaying other chunks of memory with the same address. The other gating for BANK SELECT is VMA (described above), $\overline{A14}$, $\overline{A15}$, and $\overline{EN}$ $\overline{DISP}$ $\overline{ADDR}$ (EN MPU or roughly Ø2 interval). AO and R/W from the MPU are gated at all times because pin 15 on U18 is tied to ground.

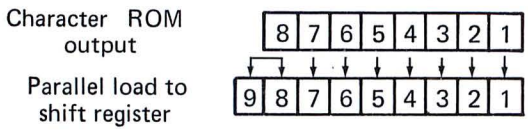
Devices U22 through U25 are MC6880 high speed bidirectional data buffers for multiplexing the Even and Odd Memory bytes into the MPU data bus ($\overline{DO}$ - $\overline{D7}$) and vice versa. In systems requiring a non-inverted data bus, MC6880 is replaced with MC6889. During an MPU read, either U22/U23 or U24/U25 is enabled by EVEN D/E or ODD D/E respectively. The buffered LSB of the MPU address, AO, determines which signal is active. During an MPU write, both U22/U23 and U24/U25 are enabled into their respective Even and Odd Memory blocks. The buffered LSB of the MPU address, AO, determines whether EVEN R/W or ODD R/W is active.

The refresh data is an even byte (EMDØ-7) and an odd byte (OMDØ-7).

Memory for Graphics Applications

Alphanumeric refresh memories are organized on a character basis. Each code stored in memory represents a 7 X 9 pattern in a 9 X 12 dot block. The dot pattern is stored in a character ROM addressed by the character code in the RAM. The repetition of a limited set of symbols on the screen to construct messages makes it possible to use a smaller amount of memory than would be required in a full graphics application where every dot is addressable as a memory location.

A "limited" graphics set of symbols for line drawings and forms is usually implemented with a special character ROM. The nine horizontal dots per character are provided by the ROM. However, most ROMS are organized by eight. It is usually acceptable to get the ninth bit from one of the eight ROM outputs, by parallel load of the 8th bit of ROM into both the eighth and ninth bits of the shift register.



A full graphics capability requires every possible dot on the screen to be stored in memory. Since the pattern is stored directly in RAM, all alphanumeric patterns are generated external to the refresh loop. Accordingly, the character ROM is placed on the MPU bus, and the dual latches drive the shift register directly. As in the alphanumeric controller, the RAM delivers two bytes (even and odd) when addressed by the VIA for refresh. Read and write addressing by the MPU is efficiently handled by bit addressing rather than byte addressing. The complete 64 K address structure of the MC6800 is decoded by hardware; only one of the eight MPU data bits is used for transfers.

A graphics terminal dedicates an MPU to the keyboard and I/O transfers with the refresh memory. All calculations (e.g. vectors), curves, rotations are done in an outboard high speed processor (e.g. microprogrammed bipolar slice). An interface between the MPU and the higher speed processor provides means for control and exchange of input parameters and results.

DISPLAY CONTROL

The DISPLAY CONTROL consists of circuitry to: sequentially access ASCII characters from the 2-PORT REFRESH MEMORY; generate character row selects; load row patterns into the Parallel-to-Serial Shift Register; serially shift the row pattern through Field Attribute circuits to the monitor as video; provide blanking, horizontal sync, and vertical sync signals; perform cursor compare and generate cursor block.

Character ROM

The purpose of the Display Control circuitry is to paint the contents of the character ROM at the designated positions on the CRT screen. Custom character ROMS contain 9 X 16 dot matrix patterns for alpha-numeric characters of various domestic and foreign fonts, limited graphics symbols, control characters, or combinations of all from the above. The addresses of the character dot matrix patterns correspond to their ASCII code representation in the case of alphanumeric and control characters. (Assignment is somewhat arbitrary for graphic symbols.)

The particular row of dots in each character dot matrix is selected by four binary row select inputs. Only 12 of the possible 16 rows are utilized in the CRT display being described. The 12 rows are adequate for shifted lower-case characters (g, j, p, q).

Referring to Figure 5 we see that a 16 K binary ROM (e.g. MCM6832) provides 128 possible ASCII characters (only 7 X 12 of the 8 X 16 dot block is used). This is a practical number of characters for most applications. Note the eighth bit of the ASCII code is always available. If it is not used for imbedded Field Attribute Codes (see FAC circuitry description), it can be used to select a second MCM6832 with a different font or graphics. The tri-state capability of the MCM6832 facilitates this mode of operation.

General Timing

All timing, including MPU Ø1 and Ø2 is derived from an MC12061 crystal oscillator running at the video rate, 17.074800 MHz. The circuit is very stable and always starts from power up. The TTL output of the MC12061 is buffered with a 74S04 before distributing the clock to avoid distortion. The distribution of the clock is critical. Video Clock and $\overline{Video\ Clock}$ are fanned out using 74S04 inverters in the same package (for minimum differential propagation delay). Loads are split equally. An alternate distributor is a high speed clock driver with complementary outputs. An important consideration is to keep all 17 MHz logic close together and in proximity to a ground. This will minimize noise and distortion.

There is another method for arriving at 17.074800 MHz. Although it is more expensive, a phase-locked

Microcomputers are highly complicated devices. When you buy one you want to make sure the manufacturer has a solid reputation for reliability and support. You want to make sure he'll be in your corner a year or two down the road.

The Altair™8800 from MITS was the first general-purpose microcomputer. Today, there are more Altair computers up and running than all the other general-purpose microcomputers combined. Today, Altairs are successfully used for literally hundreds of personal, business, scientific, and industrial applications.

Because we are so popular, many people have tried to copy us. The pages of microcomputer magazines are full of advertisements for Altair compatible devices and Altair imitation computers.

Because we are NUMBER ONE, we offer a much broader range of products and services than any of our competitors. One manufacturer might be able to copy one of our computers. Another might be able to produce a working memory card. But no one can copy the overall Altair concept.

The Altair concept is a system concept aimed at practical, cost effective applications. That's why we offer three mainframes including the Altair 680b, Altair 8800a, and Altair 8800b; ten peripherals including a multi-disk system; and over 20 plug compatible modules including our new, low power 16K static memory board. That's why we are the only microcomputer manufacturer to go to the extra expense of providing our customers with quality, higher language software.

When you buy an Altair, you're not just buying a piece of equipment. You're buying years of reliable, low-cost computing. You're buying the support of the NUMBER ONE manufacturer in the microcomputer field.

2450 Alamo SE/Albuquerque, NM 87106/505-243-7821

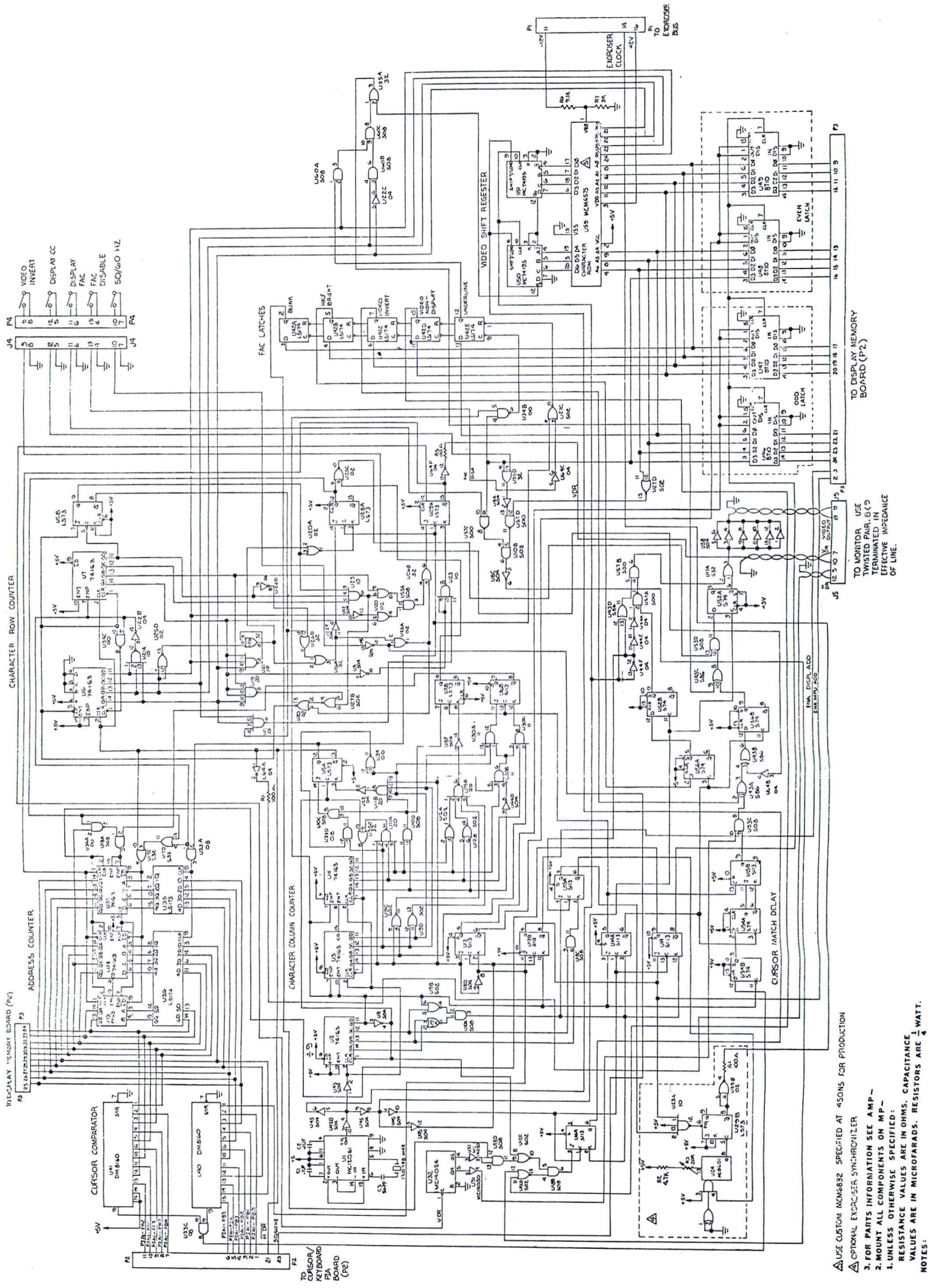
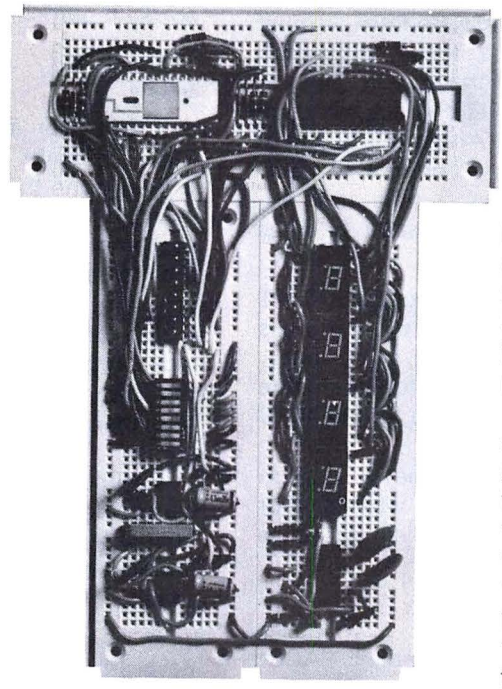


Figure 5. Schematic — Display Control Board

ANNOUNCING THREE NEW BREAKTHROUGHS IN SOLDERLESS BREADBOARDING.



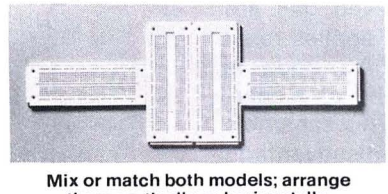
Whether you design or build circuits for fun or for profit, you owe it to yourself to discover how fast and easy CSC solderless breadboarding can be. Now, more than ever. Because of three new breakthroughs in breadboard design. And our new EXPERIMENTOR™ sockets** that make the most of them.

1. Price Who says a quality breadboard has to be expensive? For as little as $9.95, CSC's EXPERIMENTOR sockets let you design, assemble and modify circuits as fast as you can push in—or pull out—component leads. On a rugged one-piece socket with 550 solderless tie-points (94 five-point terminals *and* two 40-point bus strips).

Sockets lock together, snap apart to handle any size circuit with ease.

But don't let the low price fool you: EXPERIMENTOR sockets are precision-molded of durable, abrasion-resistant material, and feature CSC's non-corrosive, prestressed nickel-silver contacts for positive connections and longer life. All contacts are identified, too...with molded-in designations for easier circuit assembly and diagramming.

2. Compatibility CSC EXPERIMENTOR sockets end the "big-chip blues." They're the only ones with full fan-out capabilities for microprocessors and other larger DIP's, as well as 4-16-pin units. EXPERIMENTOR 600's 6/10" center is ideal for microprocessor's, clock chips, RAM's, ROM's, PROM's, etc. While EXPERIMENTOR 300's smaller 3/10" center is perfect for smaller DIP's. Both units, of course, accept transistors, LED's, resistors, capacitors, pot's—virtually all types of components with plug-in ease. As well as #22-30 solid hook-up wire for interconnections. Eliminating heat and lead damage to expensive components. And saving you more money, on parts.



Mix or match both models; arrange them vertically or horizontally.

3. Flexibility With CSC EXPERIMENTOR sockets, you can arrange your breadboard to suit your circuit...instead of vice versa. An exclusive snap-together interlocking system lets you instantly connect them. Vertically or horizontally. So you can mix or match 3/10 and 6/10" centers...expanding or contracting to meet your requirements.

CSC EXPERIMENTOR sockets can be used alone, or mounted on any convenient flat surface, thanks to molded-in mounting holes and vinyl insulation backing that prevents shorts. You can mount them with 4-40 flat-head screws from the front, or 6-32 self-tapping screws, from the rear. But however you use them, EXPERIMENTOR sockets are an unbeatable way to build and test twice the projects in half the time.

Both models feature 94 five-point terminals (vertical lines) and two 48-point bus strips (one strip shown).

WHY WAIT? CSC EXPERIMENTOR sockets are available now from your CSC dealer, or by phone from CSC, at $9.95* for the 300 and $10.95* for the 600. Call 203-624-3103 (East Coast) or 415-421-8872 (West Coast)—major credit cards are accepted.

*Manufacturer's suggested list price. Prices subject to change without notice.
U.S. Pat. No. D235,554

CONTINENTAL SPECIALTIES CORPORATION

44 Kendall Street, Box 1942, New Haven, Ct 06509
203-624-3103/TWX: 710-465-1227
West Coast office: Box 7809, San Francisco, CA 94119
415-421-8872/TWX: 910-372-7992

CIRCLE INQUIRY NO. 7

loop or phase-locked oscillator is used. The line frequency is the reference and vertical sync the comparison signal. In areas where the line frequency varies radically from nominal, the more expensive system is often required to reduce the visible beat on the CRT screen.

A ÷ 9 counter divides video rate down to a character rate of 1.9 MHz. On/Off decodes from the ÷ 9 counter are resynchronized in 74S113 high speed flip flops. The signals provide the General Timing in Figure 6.

Decodes off the Character Column Counter provide horizontal timing (Figure 7). Vertical timing is from the Character Row Counter (Figure 8).

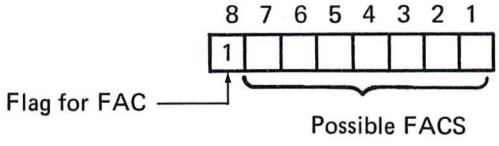
Note the operation of the Address Counter. It must repeat each character address 12 times to paint a complete line of 80 characters on the screen. This requires storing the address of the first character in each line (function of the 74LS latches). Another function of the Address Counter is to advance to address 64 during V blank. (Effectively, this amounts to a start address of 128 since the memory is addressed two bytes at a time.)

Another function of the Display Control Electronics is cursor compare. The contents of the Address Counter are compared with PIA data for coincidence. The DM8160's are exclusive-or comparators. PIA data is a binary address which is manipulated by the MPU for cursor control.

The other circuitry in the Display Control portion generates invert/non-invert cursor block when cursor compare is sensed and furnishes FAC (Field Attribute Code) logic.

FACS (Field Attribute Codes)

There are two popular methods for handling FACS. In the wide memory method, the memory size is increased by adding bits to each character in memory. Each bit controls a different attribute code for that character. The other method imbeds FAC characters in refresh memory. The eighth bit of the ASCII code is usually decoded as a FAC flag.



When it is a logical one, the other seven bits are latched as FACS. Once latched, the FAC applies to all subsequent characters until another FAC code is decoded. The only exception is at the end of a character line; all FACS are hardware reset. This scheme's advantage is low cost to implement; the disadvantage is the use of a memory location per FAC code. Not only is character density decreased, but the individual characters in a string can not be accented with FACS. It is possible to get around this drawback by stripping FACS from the memory before display, but requires extra hardware and is a programming nightmare. When individually accented characters are required, it is usually better to implement a wider memory.

The wide memory approach to FACS may seem clumsy at first glance . . . the MPU has an 8 bit bus. How are 8 bit MPU transfers done? Construct two pages of memory — a page of ASCII characters 2K X 8, and a page of attributes (2K X 1, 2, 3, 4, . . ., 8). The attribute page is a mask and need only be accessed when the attribute changes or must be read.

For simplicity, the imbedded FAC method was implemented in the CRT terminal. Few changes in Display Control circuitry are required for a wide memory approach.

CURSOR/KEYBOARD

Referring to Figure 9 we can see how to add a cursor/keyboard interface to this "glass teletype."

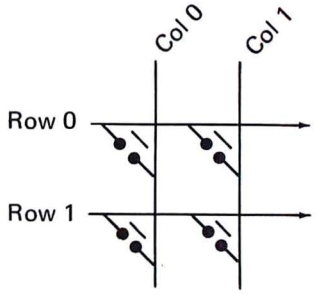
Cursor

The cursor address is stored as the contents of a PIA. The eight least significant binary bits are PB0 - PB7 and the three most significant binary bits are PA0 - PA2. There is a one-to-one correspondence between the binary cursor address and a memory location on the screen. The assignment of bits in the PIA is for programming convenience. An STX instruction to the A side of the PIA writes the higher order byte of the index register into the A side of the cursor PIA and the lower order byte into the B side of the cursor PIA (provided the address lines to RS0 and RS1 are reversed).

The cursor address is binary rather than X-Y because the binary address manipulations are more frequent. When X-Y addressing is required (e.g. communication interface), a conversion subroutine is called.

Keyboard

Either a fully-encoded or non-encoded keyboard can be designed. Whichever, a PIA provides the interface to the MPU. In a fully-encoded keyboard, the keyboard hardware generates a strobe and an ASCII encoded character corresponding to the depressed key. All debounce is handled by the keyboard hardware. The strobe pulse applied to the PIA CA or CB inputs causes an interrupt to the MPU. The MPU reads the ASCII character through the PIA and performs the appropriate function. For an alphanumeric character, the MPU writes the data at the present cursor location and increments cursor PIA contents. For control characters, the corresponding commands are executed. (e.g. space, carriage return, insert, delete, etc.)



A non-encoded keyboard is a set of switches wired in a column/row matrix.

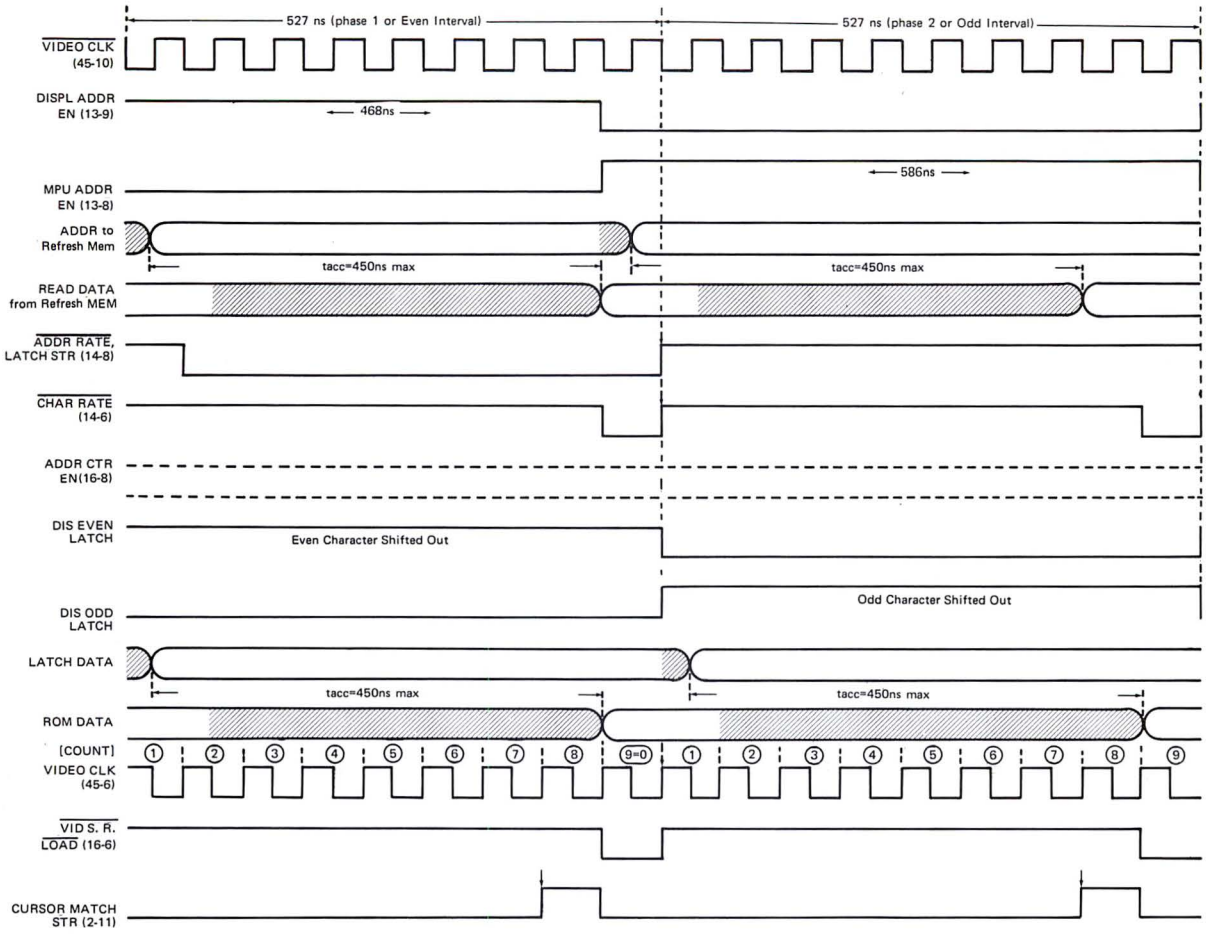


Figure 6. General Timing

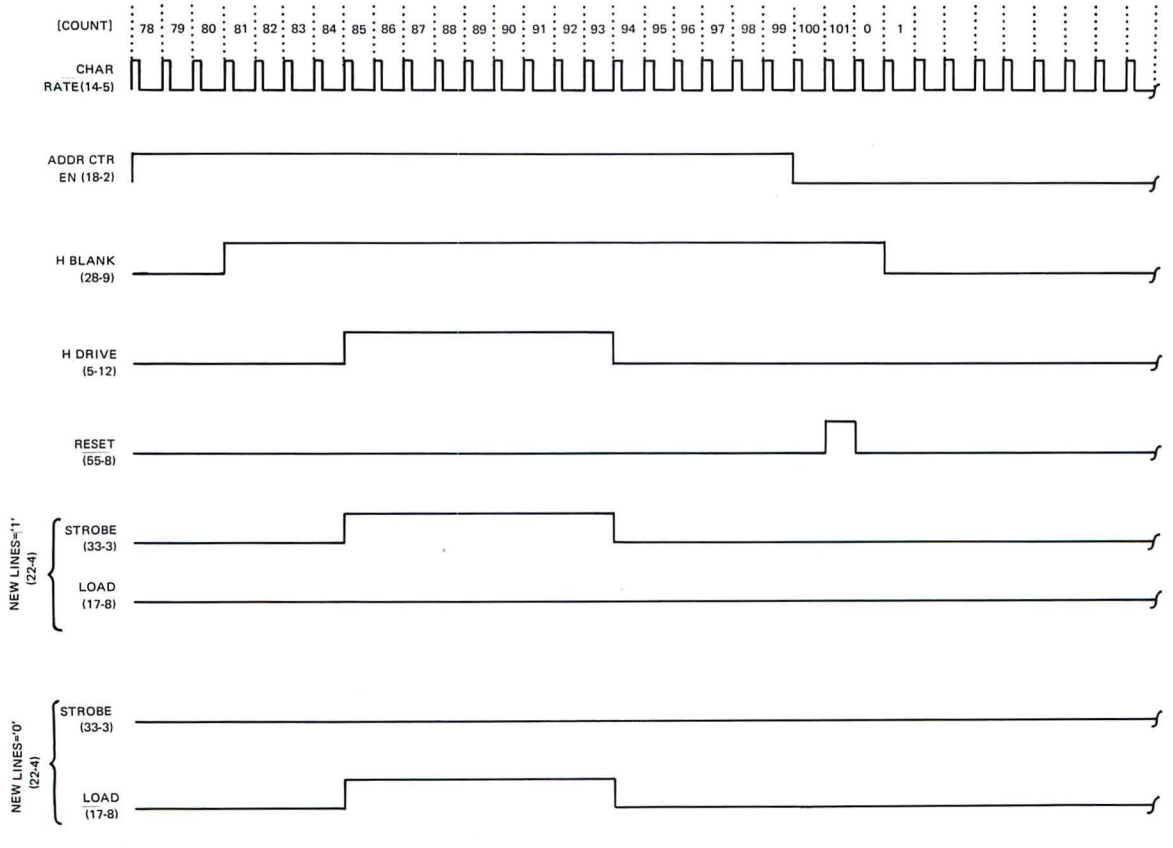


Figure 7. Horizontal Timing

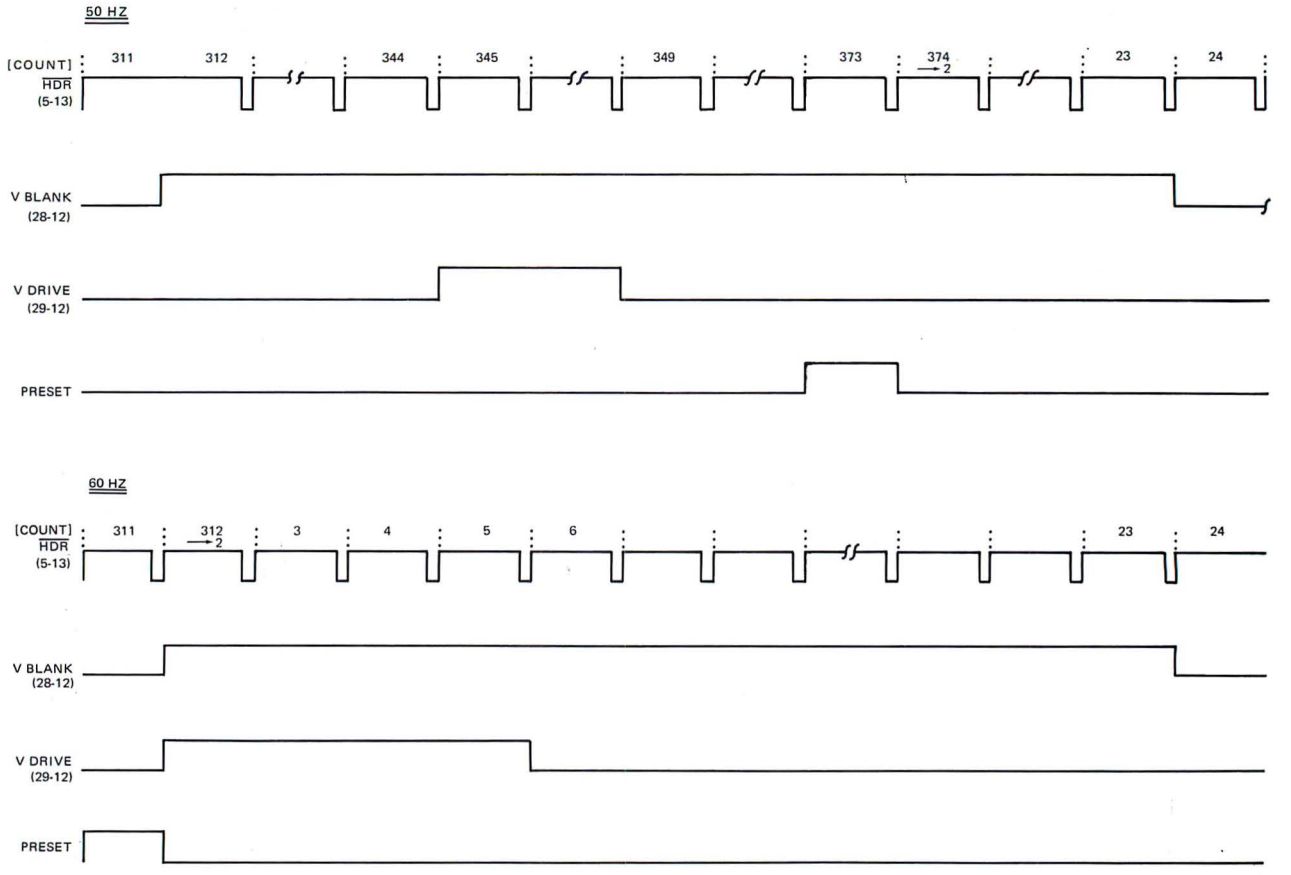
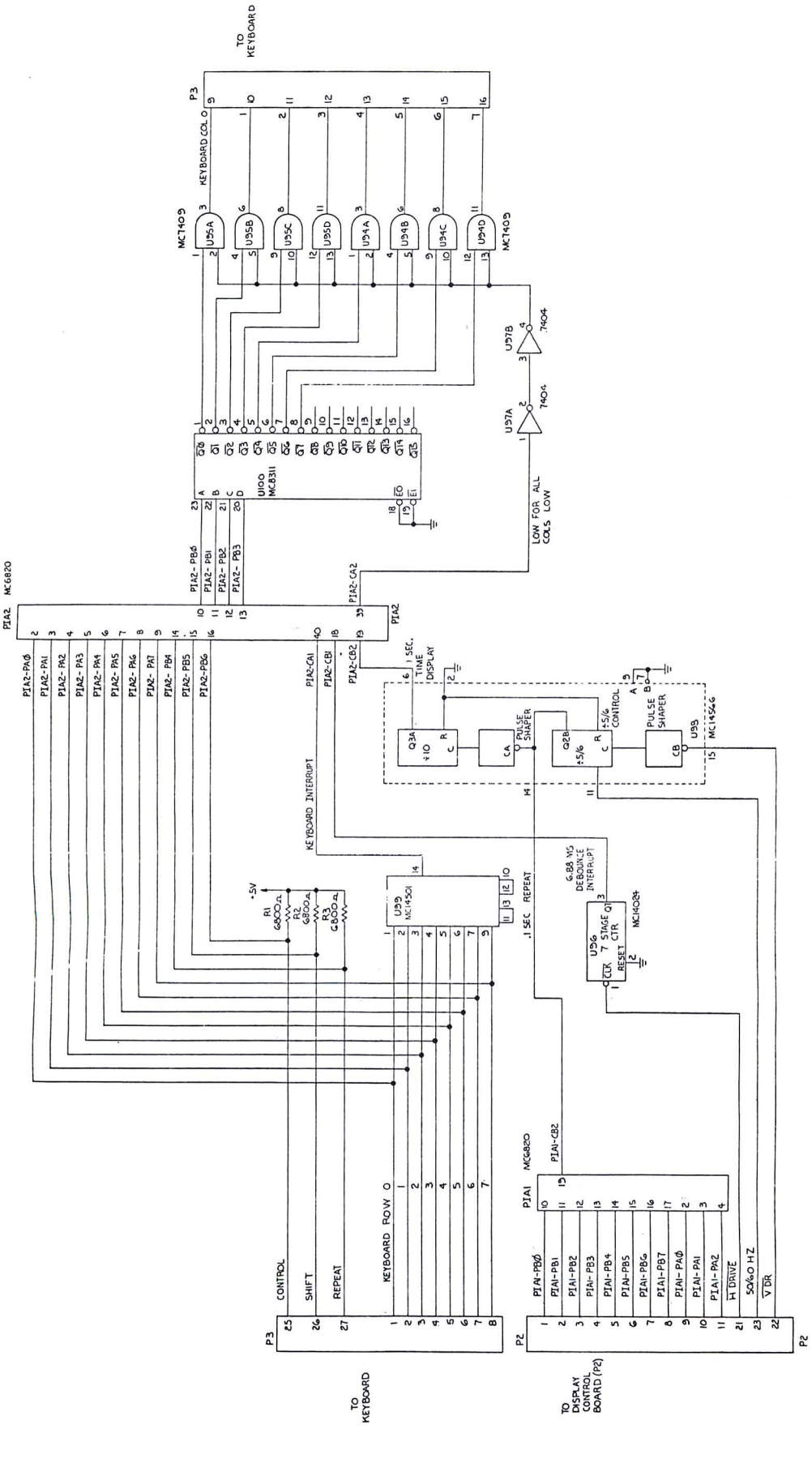


Figure 8. Vertical Timing



NOTES:
1. UNLESS OTHERWISE SPECIFIED:
   RESISTANCE VALUES ARE IN OHMS, CAPACITANCE VALUES ARE IN MICROFARADS. RESISTORS ARE $\frac{1}{4}$ WATT.
2. MOUNT ALL COMPONENTS ON MP-
3. FOR PARTS INFORMATION SEE AMP-
4. ADDITION TO MEX6820 I/O MODULE

Figure 9. Cursor/Keyboard PIA Board

If the columns are scanned one at a time, and the rows read back, the simultaneous depression of any one or two keys can be discerned. For n keys, diodes are wired in series with the switch contacts.

The scanning of columns, reading of rows, and switch debounce are under software control. Keys are strategically placed in the matrix so the column and row location easily translate into an ASCII code. Cost savings and flexibility of this non-encoded keyboard versus a fully-encoded keyboard sometimes justifies the additional MPU overhead in a basic CRT terminal.

A non-encoded approach is described here. Referring to the schematic, the keyboard columns are normally held low. When key(s) are depressed, lows appear on the keyboard row inputs. A keyboard interrupt is generated, which starts a scan. Keyboard column outputs are brought low in sequential order. Together with the Shift and Control PIA inputs, the active columns and rows are encoded as an ASCII character and temporarily stored. A 6.88 msec interrupt is provided for debounce and a PIA input for REPEAT (also .1 sec interrupt for this function and a 1 second interrupt for clock functions).

The software for a non-encoded keyboard ranges from simple to quite complex depending on how foolproof the algorithm is.

Comparison of CRT Terminal Architectures

The central design criteria of a modern CRT terminal is the method used for multiplexing refresh memory between the MPU and the display refresh circuitry. In this section we will discuss time-division multiplexing and priority multiplexing. Throughput versus hardware complexity for different techniques will be analyzed. Those systems which result in operator annoyance in many applications (e.g. Key-to-Disk), and are not appropriate for the modern CRT terminal.

Priority Multiplexing of Refresh Memory

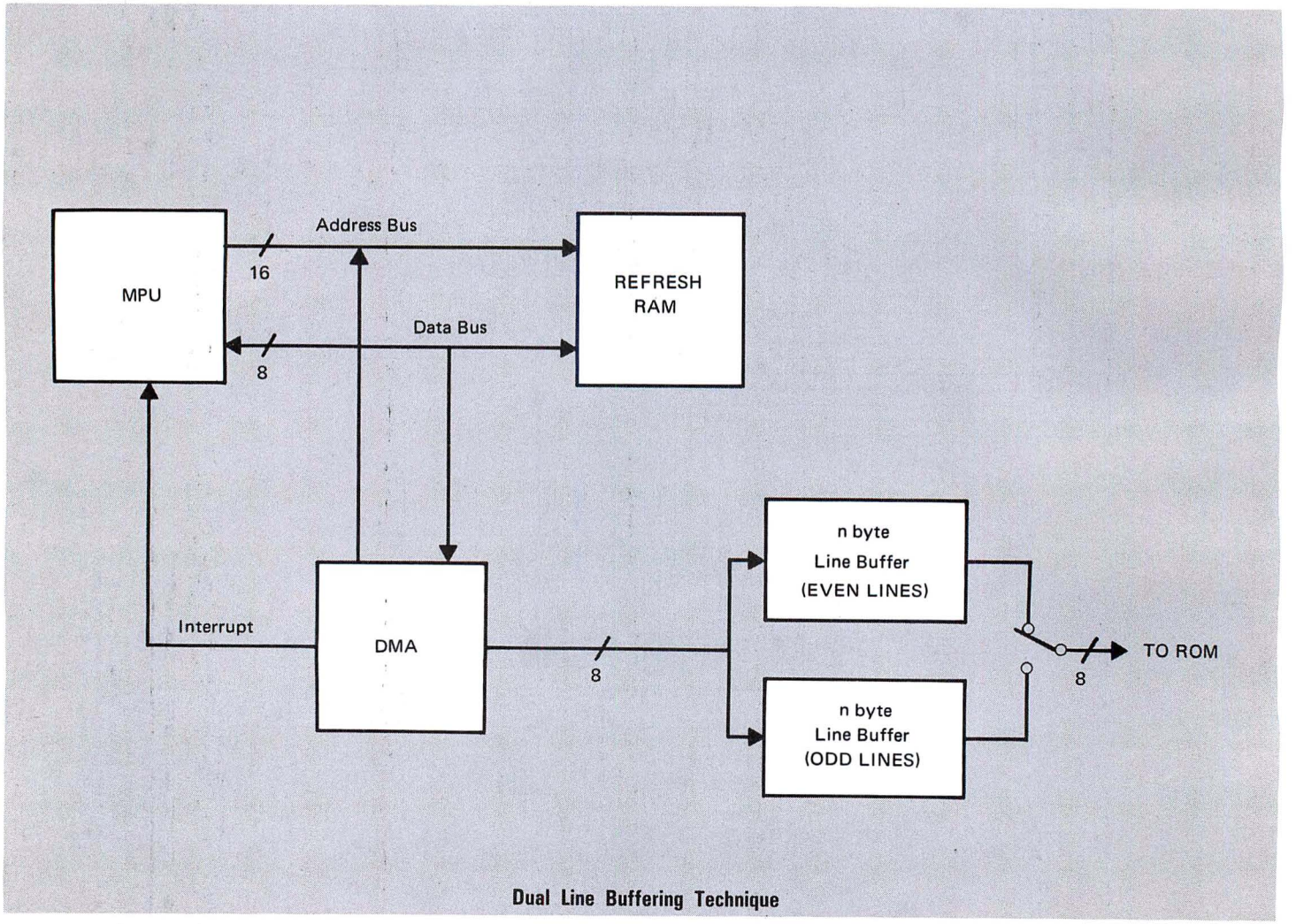
These techniques fall into two general categories:

A. MPU grabs Refresh Memory and locks out Refresh Circuitry.

B. Refresh circuitry grabs Refresh Memory and locks out MPU.

Method A results in zero burden on the MPU; however, a sufficient number of refresh characters must be pipe-lined into FIFO to keep the screen refreshed during an MPU access. The fast memory and complex hardware to accomplish this task is unattractive.

Method B is probably the most widespread technique in use today. The dual line buffer approach is



Dual Line Buffering Technique

DIGITAL DATA RECORDERS

USING 3M DATA CARTRIDGES

MODEL 3M3

199.95

BRAND NEW DESIGN

Featuring the radically new "UNIBOARD" method of construction of data cartridge drives. The major computer manufacturers are changing from cassettes to cartridges at a rapid pace because of freedom from binding and greater data reliability. Now, these professional type units are priced within the range of all data users. Being made primarily as OEM data storage units for the world's major manufacturers, these units, together with controller board and software ROM, are being made available to the individual user as well.

*Appearance and specifications may be changed slightly following acceptance tests now being conducted by OEM users.

MODEL 3M1

169.95

MODEL 3M3—Uses the 3M Data Cartridge, Model DC300. This cartridge contains 300 feet of .250 tape in a sealed container. Records and plays at 9600 baud NRZ, 4800 baud P.E. Nominal speed 8" per second. Max. recommended flux density 1200 fcpi. Using four tracks, you can store nearly 2 megabytes of data on a cartridge. Cartridge measures 4" by 6". Turns counter indicates tape position. Inter-record gap light gives more accurate position 2SIO(R) is *not* required for use, but is highly recommended for 8080 and Z80 systems.

COMMON SPECIFICATIONS: FULL SOFTWARE CONTROL of record, play, fast forward and rewind. LED indicates inter-record gaps. EOT and BOT are sensed and automatically shut down recorder. Can also be manually operated using the switches on top which parallel the software control signals when not under software control. Signal feedback makes it possible to software search for inter-record gaps at high speed. 117 V— 50 Hz— 5 watts.

TWO I/O PORT CONTROLLER WITH ROM—Controls your terminal and one or two cassettes or cartridge units. On board ROM (For 8080 and Z80) has terminal and cassette software for turn on and go operation. NO MORE BOOTSTRAPPING. Plug in compatible with Altair and IMSAI. Loads and Dumps memory in Hex from the keyboard, formats tape files, punches tape, functions as a word processor and searches for files and four letter strings within files. Keyboard controls the cartridge units above on rewind and fast forward. Special keyboard codes enable you to dump and read Phase Encoded tapes as well as NRZ tapes. (Including K.C. Std.) Call routines give access to these from your software.

MODEL 2SIO(R) With 1 ROM for NRZ Cassettes $169.95
Assembled & Tested) (Half of above Program)
With 2 ROM's for Data Cartridges and P.E. cassettes.
$189.95 (Full Program)
Kits available for $30 off above prices.

OVERSEAS: EXPORT VERSION — 220 V—50 Hz. Write Factory or— MEGATRON, 8011 Putzbrunn, Munchen, Germany; Nippon Automation 5-16-7 Shiba, Minato-Ku, Tokyo; EBASA, Enrique Barges, 7 Barcelona, Spain; Hobby Data, SpireaVagen 5, Malmo, Sweden; G. Ashbee, 172 Ifield Road, London SW 10-9AG.

MODEL 3M1—Uses the 3M Data Cartridge type DC100A. This cartridge contains 150 feet of .150 tape and is the same cartridge used by H.P. and others. Runs at 4800 baud NRZ, 2400 baud P.E. Tape speed adjustable, but nominally set at 5"/second. Maximum recommended flux density 1200 fcpi. Cartridge measures 2-1/8" by 3-1/4". This model is ultra compact, yet extremely capable. It is intended for word processing, mailing list use and other applications requiring the compact storage of data. Data location is by inter-record gaps and automatic file search. See Common Specs and 2SIO(R). 2SIO(R) is *not* required for use, but is highly recommended for 8080 and Z80 users.

For 8080 and Z80 users: Comes complete with software program listings for the programs on the 2SIO(R) ROM. 6800 software is being written, but not yet completed. These programs give *FULL SOFTWARE CONTROL.*

CARTRIDGE AVAILABILITY: Cartridges are made by 3M, ITC, Wabash and others. They are available at all computer supply houses and most major computer service centers. We can also supply them at normal current list prices.

NEW: AUDIO CASSETTE INTERFACE* Phase Encoding interface for use with audio cassettes or NRZ recorders. Runs 2400 baud phase encoded on good quality audio cassette recorders. May also be used with 2SIO(R) to use the 2SIO(R) cassette programs with your audio cassette player. Can also accommodate "Tarbell" tapes and K.C. Std. tapes.
$50.00, Wired & Tested.
$35.00, Kit Form.

***NOTE: You do not require an interface with the 3M1 and 3M3 unless you Phase Encode. But, you do need an interface to use the 2SIO(R) with your own audio cassette.**

"COMPUTER AID" and "UNIBOARD" are trademarks of the NATIONAL MULTIPLEX CORPORATION. The 3M Data Cartridges are covered by 3M Patents and Marks. "UNIBOARD" Patents Pending.

For U.P.S. delivery, add $2.00 each item. Overseas and air shipments charges collect. N.J. Residents add 5% Sales Tax. WRITE or CALL for further information. Phone Orders on Master Charge and Bank-Americard accepted. MC BA

NATIONAL MULTIPLEX CORPORATION
3474 Rand Avenue, South Plainfield, N.J. 07080, Box 288.
Phone (201) 561-3600 TWX 710-997-9530

CIRCLE INQUIRY NO. 21

representative of this class. All data for a line on the screen is stored in FIFO's (usually 80 bytes).

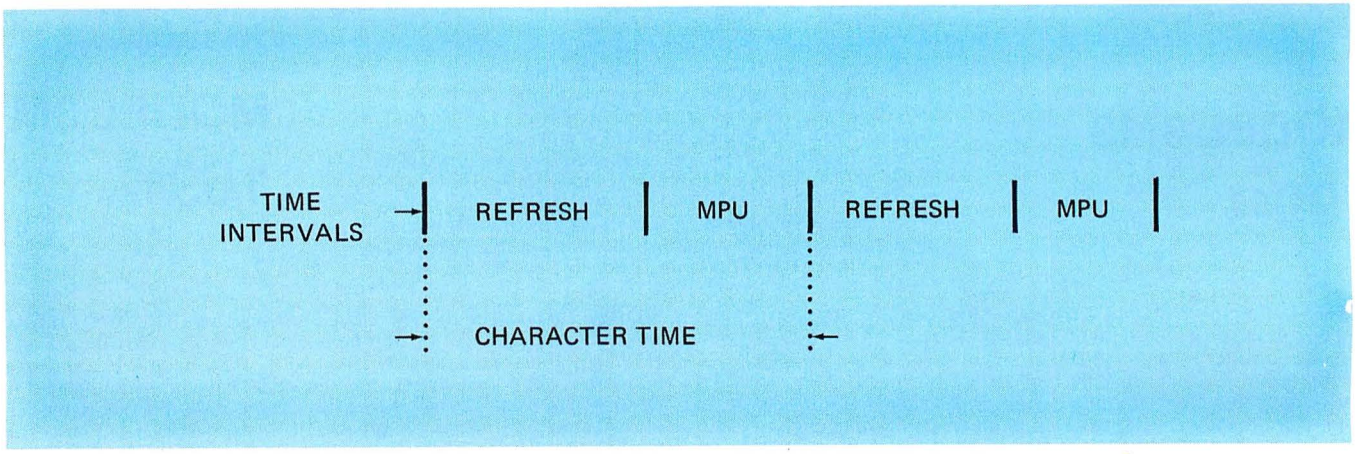
While the Odd line buffer is keeping the display refreshed, the Even line buffer is being filled with the next displayed line from Refresh RAM through a DMA channel. The functions are alternately reversed. In an alphanumeric terminal, average MPU burden is between 15 and 30% with peak burden on the MPU of 100% for 80-100 microseconds when loading either line buffer. The MPU is stalled during this transfer and cannot service high speed interfaces (e.g. Floppy Disk). With the addition of extra hardware, the Refresh page of the RAM is isolated from the processor bus during DMA, thus allowing the MPU to continue processing provided it attempts no accesses to memory. In conclusion, the dual-line buffers and DMA configuration is more expensive, has higher parts count, and imposes a severe burden on the MPU compared to the time-division multiplexed technique.

Furthermore, the DMA approach is very inefficient for full graphics, since there are no recirculations of the line buffer as in an alphanumeric display.

refreshed during an MPU synchronization delay. The FIFO is typically two bytes deep. The disadvantage of this technique is the requirement for fast memory. It is considerably simplified if the time-division multiplexing is made synchronous with character rate. Sync delay is no longer required for refresh and the FIFO circuitry is eliminated (a single byte latch is still required). Also memory speed is reduced. This simplification is not always acceptable; e.g. it may result in excessive MPU "cycle-stretching" at slow character rates.

The technique for sharing CRT Refresh Memory described in this article is a special case of time-division multiplexing. The access intervals for Refresh and MPU are 01 and 02, respectively.

As 01 and 02 clocking signals are outputted even during wait states, no matter how long this MPU is forced into a wait state the screen will remain refreshed. Since there is no overlap of accesses, no contention circuitry is required and the MPU burden is zero at all times. While you may not want or need to duplicate the entire circuitry described in this article, the bi-phase memory access technique may be used

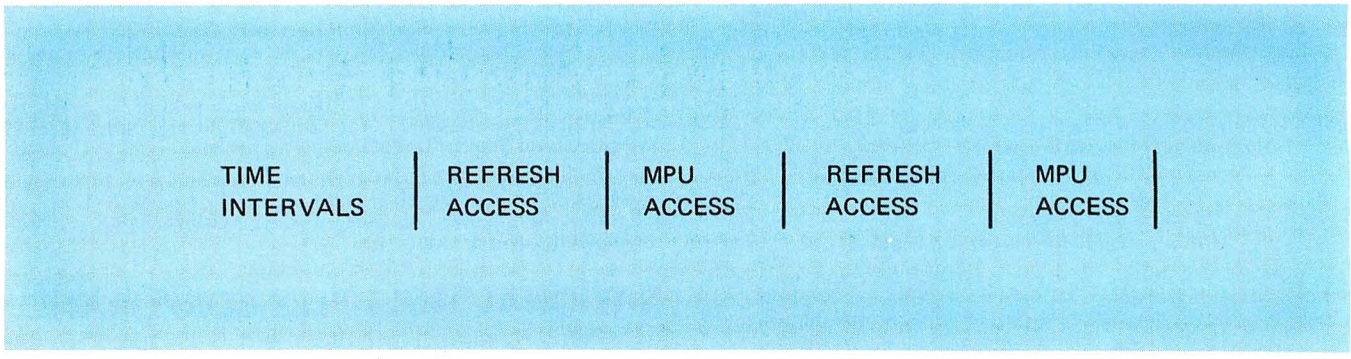


Time Division Multiplexing of Refresh Memory

In the most general time-division multiplexing situation, either an MPU or Refresh access may commence or end asynchronously to time-division multiplexing intervals. By delaying accesses, the MPU and Refresh circuitry are brought in sync with their respective time intervals. The sync delay for MPU access, at the expense of throughput, is implemented by "cycle-stretching." The sync delay for Refresh access is accomplished by pipe lining a sufficient number of refresh characters into a FIFO to keep the screen

wherever it is necessary to use a DMA (Direct Memory Access) for two major systems.

The M6800 implimentation is very simple due to constant cycle lengths, whereas it is difficult if not impossible with variable cycle length MPU's. The only drawback to this scheme is that the MPU rate is a derivative of the character rate. This translates into a clock rate lower than the maximum MPU clock rate allowable. Consequently, the throughput is reduced. For example, the screen density in this article reduces the throughput by 5% because the MPU runs at 950 KHz.



The Compucolor 8001 System.

It's A Stand Alone Micro Computer With Color Input/Output Capabilities All In One Package. For Only $2995.

If you're looking for an input device, an output device and a micro computer all in one package, you've found it. The Compucolor 8001. It's here now, in color, on sale for only $2995.

We gave it a memory of its own.

And Floppy Tape Memory is just for starters. Look at these other features. BASIC Language, 8080 CPU, 8 color CRT Terminal, 8K RAM Workspace, Selectable Baud Rate to 9600, Two RS 232 I/O's, Keyboard with 16 Function Keys, Background Color, Lower Case ASCII Characters, Roll, Insert/Delete, 48 Line X 80 Characters/Line, 2X Character Height, thorough operating instructions and a Graphics Mode with 160 X 192 Elements. And our unique Nine Sector Convergence System guarantees you quick set-up, exceptional stability and outstanding color registration in three to five minutes. If you can find a better buy in a color Intelligent CRT and Micro Computer system, let us know. We think we've got the best of both worlds at the best price going. And we want to prove it to you.

Name your game.

After all, you'll have your very own personal computer right at your fingertips. For the most simple or complex tasks. Or just plain fun. The applications are unlimited. Color graphics and computations, check book balancing, educational instruction, tutoring and a unique variety of computer games. Like Star Trek and Hangman and Pong. You can even sit back and enjoy a game of chess. Like we said, the applications are unlimited.

How about a little demonstration?

You'll find a list of our distributors at the bottom of the page. So drop by and ask for a demonstration. Get some answers to your questions. And if you aren't near one of our distributors, give us a call. We've got the answers. The Compucolor 8001. You won't find a better buy in a color CRT Terminal and Micro Computer.

Compucolor Corporation
A subsidiary of
Intelligent Systems Corp.®

5965 Peachtree Corners East
Norcross, Georgia 30071
Telephone (404) 449-5961

CALIFORNIA
Byte Shop
155 Blossom Hill Rd.
San Jose, Ca. 95123
Larry Grihalva
(408) 226-8383

Computer Store
1093 Mission St.
San Francisco, Ca. 94103
Al Chern
(415) 431-0640

CALIFORNIA
The Computer Center
8205 Ronson Rd.
San Diego, Ca. 92111
Ron Eate
(714) 292-5302

The Computer Mart of Los Angeles
625 W. Katella No. 10
Orange, Ca. 92667
George Tate
(714) 633-1222

GEORGIA
The Computer Systems Center
3330 Piedmont Rd., NE
Atlanta, Ga. 30305
Jim Dunion
(404) 231-1691

ILLINOIS
Itty Bitty Machine
1316 Chicago Ave.
Evanston, Ill. 60201
Jim Bannish
(312) 328-6800

INDIANA
Home Computer Shop
10447 Chris Dr.
Indianapolis, Ind. 46229
James B. Baughn
(317) 894-3319

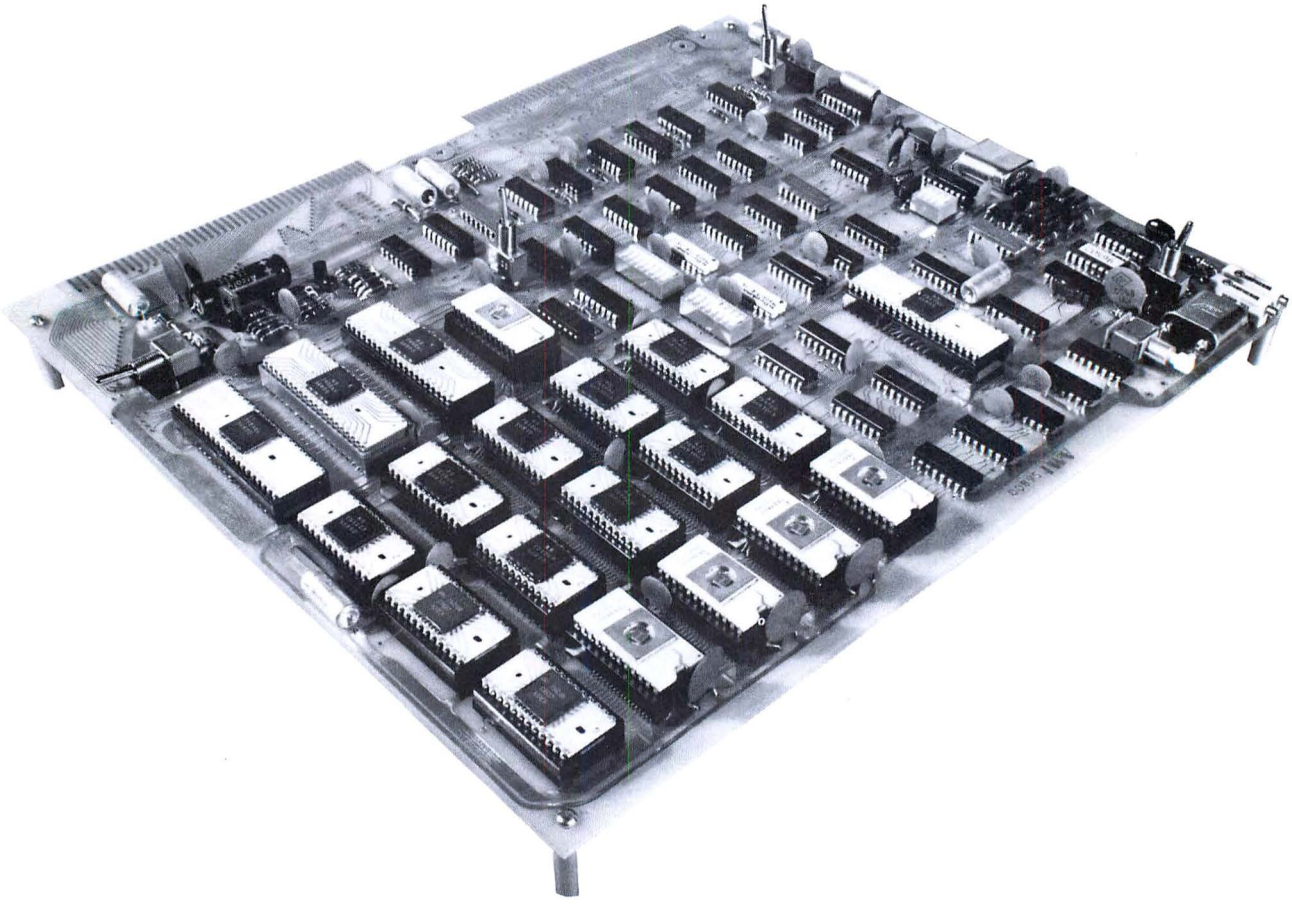
MASSACHUSETTS
The Computer Store
120 Cambridge St.
Burlington, Mass. 01803
Sid Halligan
(617) 272-8770

WASHINGTON
Retail Computer Store
410 N.E. 72nd Street
Seattle, Wash. 98115
Tim Broom
(206) 524-4101

Or Contact Us Direct

5965 Peachtree Corners East
Norcross, Georgia 30071
Telephone (404) 449-5961

AMI's EVK Series Microcomputer Prototyping Boards



By Robert A. Stevens

INTRODUCTION

This article is part #2 of a series of articles on the EVK Microcomputer hardware, firmware and supporting software. This month's article covers the EVK Microcomputer board architecture while last month's article described the functional architecture and characteristics of AMI's Microcomputer IC chip set.

EVK CONFIGURATIONS — The AMI EVK Microcomputer is a single board microcomputer mechanized with a standard S6800 MPU. The EVK Microcomputer comes in four basic configurations; EVK99, EVK100, EVK200, & EVK300, all of which use the same 10½" x 12" printed circuit board. EVK99 is a kit that includes the PCB and Microcomputer ICs consisting of one 6800 MPU, four 6810 RAM's, one 6820 PIA, two 6830 ROM, and one 6850 ACIA. EVK100 & EVK200 are kit configurations that include PCB, Microcomputer & T^2L IC's and differ from each other by the amount of hardware, memory and firmware (software in ROM) included with each configuration. EVK300 is the EVK200 kit with more EPROM memory and is factory assembled and tested. A Tiny BASIC Interpreter program is also available at no charge for the EVK300 Microcomputer board. Table 1, EVK Microcomputer Configuration Summary, shows the comparison between the different EVK configurations.

EVK BOARD CHARACTERISTICS	EVK 99	EVK 100	EVK200	EVK300
CPU	S6800	S6800	S6800	S6800
WORD SIZE	8 BITS	8 BITS	8 BITS	8 BITS
ADDRESS BUS	16 BITS (64K)	16 BITS (64K)	16 BITS (64K)	16 BITS (64K)
ROM	2K BYTES S6831 ROM	2K BYTES S6831 ROM	2K BYTES S6831 ROM	2K BYTES S6831 ROM
EPROM – VIRGIN	———	———	512 BYTES S6834 EPROM	2K BYTES S6834 EPROM
STATIC RAM	512 BYTES S6810 RAM	512 BYTES S6810 RAM	1K BYTES S6810 RAM	1K BYTES S6810 RAM
EPROM PROGRAMMING	———	———	PROGRAMS S6834 EPROM's	PROGRAMS S6834 EPROM's
I/O PORTS	1 PIA=2 PORTS – 8 BITS/PORT	———	3 PIA's=6 PORTS 8 BITS/PORT	3 PIA's=6 PORTS 8 BITS/PORT
ASR 33/35 TTY SERIAL INTERFACE	ACIA S6850	ACIA S6850 WITH 20ma CURRENT LOOP	ACIA S6850 WITH 20ma CURRNET LOOP	ACIA S6850 WITH 20 ma CURRENT LOOP
RS232C EIA SERIAL INTERFACE	ACIA S6850	ACIA S6800	ACIA S6850 WITH EIA RS232C	ACIA S6850 WITH EIA RS232C
INTERVAL TIMER (CRYSTAL)	———	———	1 ms & 100 μs TIME INTERVALS	1 ms & 100 μs TIME INTERVALS
MPU CRYSTAL CLOCK			INCLUDED	INCLUDED
CLOCK OUTPUTS (CRYSTAL)	———	16X BAUD RATE	2.4576 MHz, 1 MHz & 16X BAUD RATE	2.4576 MHz, 1 MHz & 16X BAUD RATE
DMA MODES	———	———	HALT MPU MODE, CYCLE STEAL MODE & MUX MODE	HALT MPU MODE, CYCLE STEAL MODE & MUX MODE
RESTART ADDRESS SELECTION	———	TWO 8 BIT DIP TOGGLE SWITCHES	TWO 8 BIT DIP TOGGLE SWITCHES	TWO 8 BIT DIP TOGGLE SWITCHES
TTY MONITOR SOFTWARE	PROTO ROM RESIDENT	PROTO ROM RESIDENT	PROTO ROM RESIDENT	PROTO ROM RESIDENT
SUBROUTINE PROGRAM LIBRARY SOFTWARE	RS^3 ROM RESIDENT	RS^3 ROM RESIDENT	RS^3 ROM RESIDENT	RS^3 ROM RESIDENT
ROM RESIDENT ASSEMBLER	$35.00 OPTION	$35.00 OPTION	$35.00 OPTION	$35.00 OPTION
OEM SINGLE QUANTITY PRICE	$133.00	$295.00	$495.00	$765.00

TABLE 1 EVK MICROCOMPUTER CONFIGURATION SUMMARY

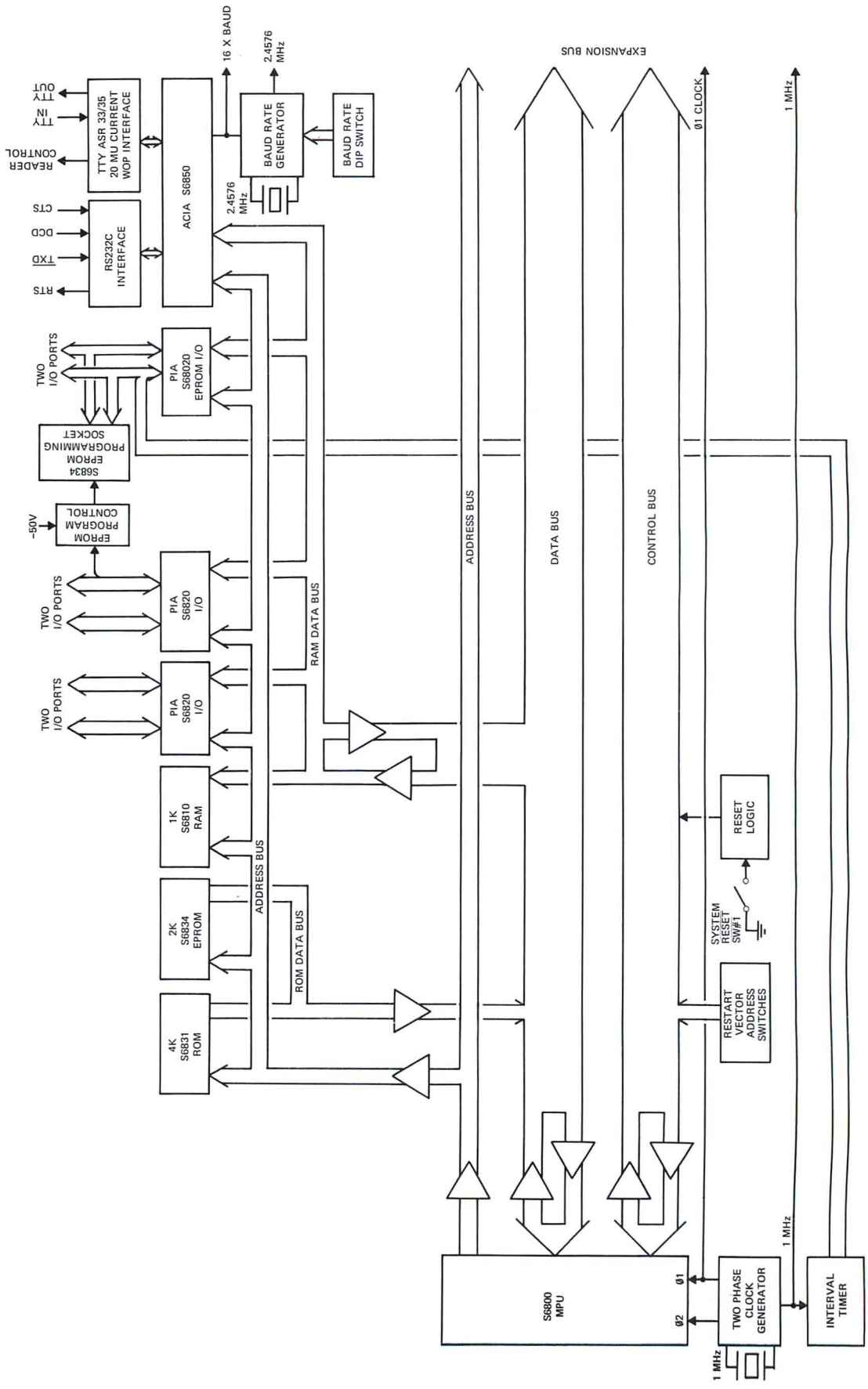
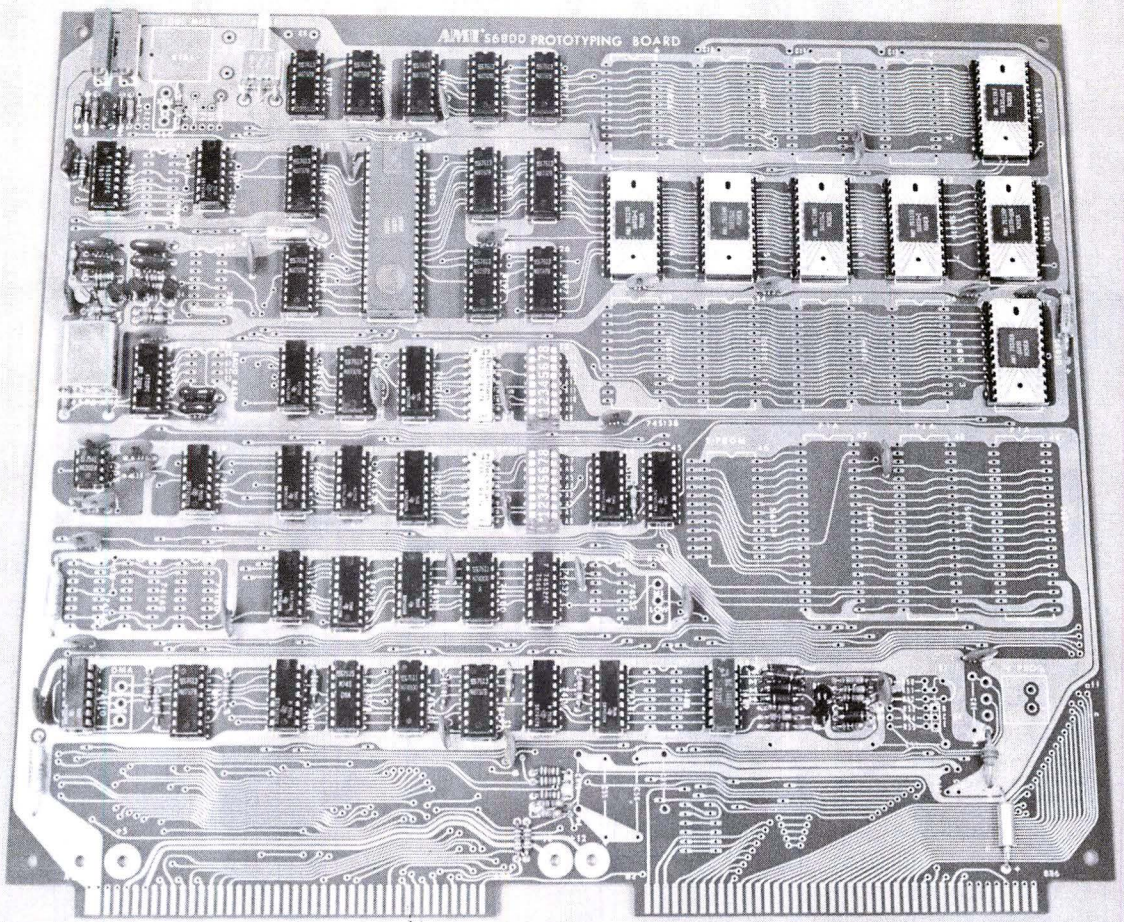


Figure 1. EVK Series Microcomputer Board



MAJOR EVK MICROCOMPUTER FEATURES

The common denominator EVK Microcomputer PCB provides the following on board major features when fully populated with hardware and software including options;

- 4K Bytes S6831 ROM memory (2K using S6831 ROM's)
- 2K Bytes S6834 EPROM memory
- 1 K Bytes S6810 Static RAM memory
- On board S6834 EPROM programming
- Six 8 bit PTA I/O ports
- 20 ma serial TTY current loop port interface
- RS232C EIA serial I/O port
- Switch selected baud rates to 19,200 bauds
- 1 MHz crystal or variable one shot MPU clock
- 5 crystal controlled timing signals available at PCB interface
  (2.4756 MHz 1MHz 16x baud rate, 100 μs & 1 ms) ms)
- Interrupt internal timing (100μs & 1 ms)
- Switch selectable MPU restart address
- 200 ms Power On Reset delay
- 3 DMA modes (HALT MPU, CYCLE STEAL & MUX)
- TTY PROTO Monitor System resident in ROM
- RS^3 ROM Subroutine Library resident in ROM
- ROM Resident Assembler — option
- Up to 40 ma @ 0.4V external bus loading
- 8T97 three state MPU bus drivers
- All MPU signal lines isolated & buffered
- System expansion via two 86 pin connectors

TYPICAL EVK MICROCOMPUTER APPLICATIONS

The EVK Microcomputer board allows the hardware development engineer, the logic designer, the programmer, the systems engineer, the mathematician, the scientist, the chemist or the hobbyist to have a complete working Microcomputer system, including development software by adding a low cost power supply and an ASR 33 TTY to the EVK Microcomputer board. The EVK series of Microcomputers boards allows the owner/user to use one of these boards to:

- Evaluate the complete set of AMI's family of Microcomputer IC's at a low investment of time & money — no design time is required.
- Serve as a general purpose Microcomputer for low volume systems to which the systems engineer can easily add additional I/O ports and memory.
- Serve as a low cost quick turn around prototype system to evaluate total system mechanization concept (hardware & software) and market acceptance prior to committing to a custom design system for large volume production.
- Serve as a low cost minimal 6800 Microcomputer application software development system.
- Serve as a low cost general purpose Microcomputer to run numerous application software programs.

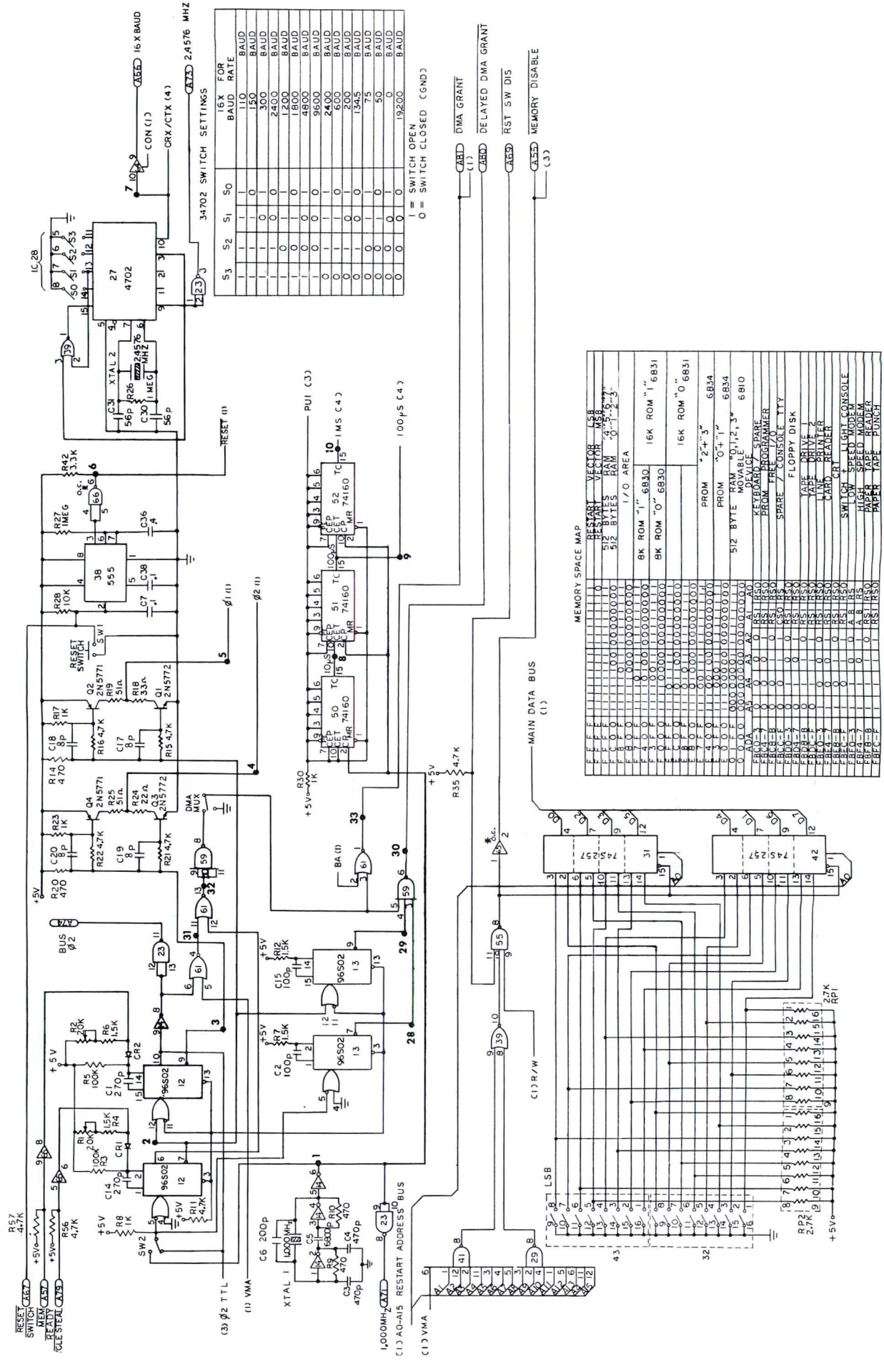


Figure 2. AMI S6800 Prototyping Board Schematic Diagram

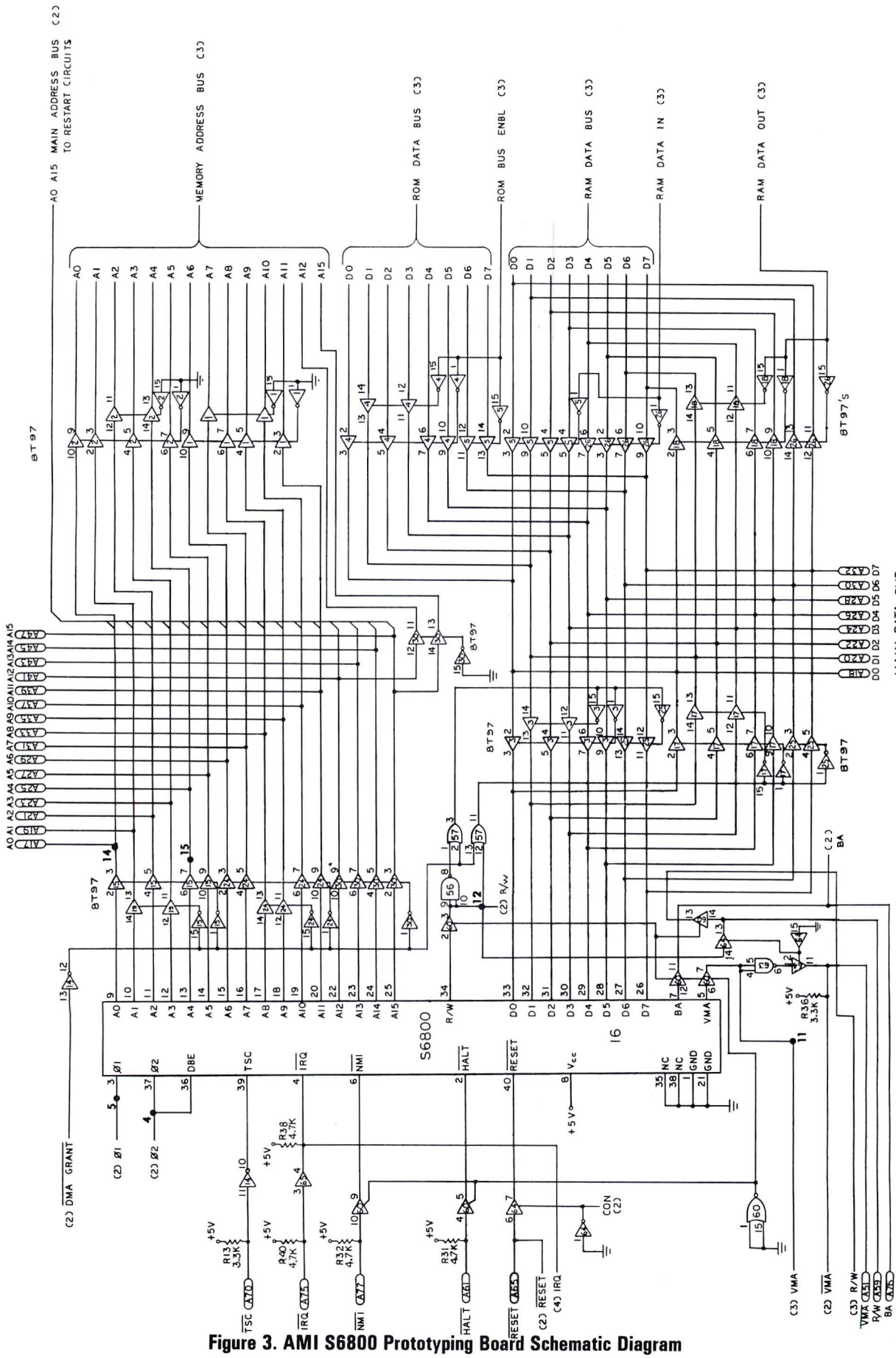


Figure 3. AMI S6800 Prototyping Board Schematic Diagram

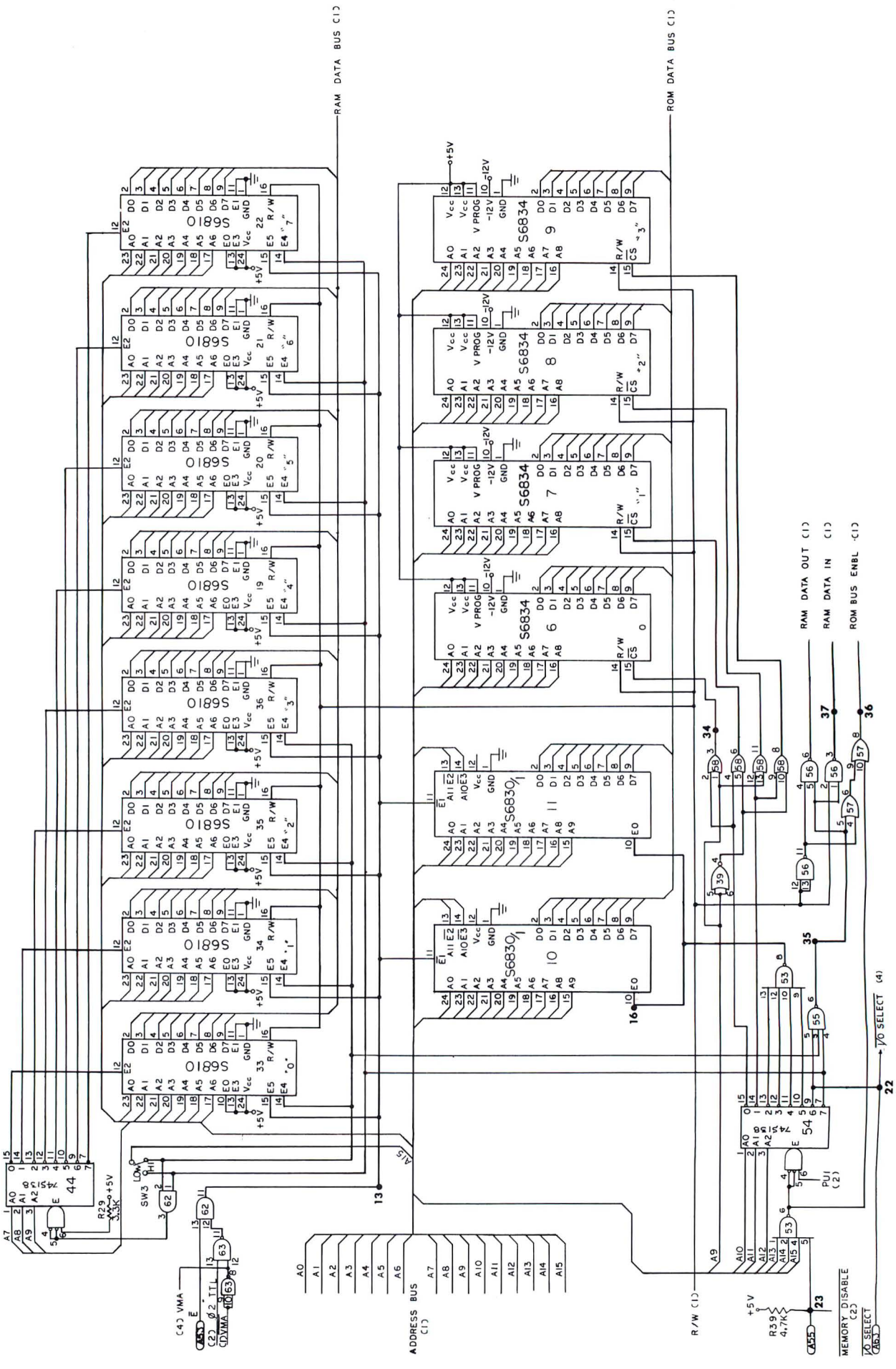


Figure 4. AMI S6800 Prototyping Board Schematic Diagram

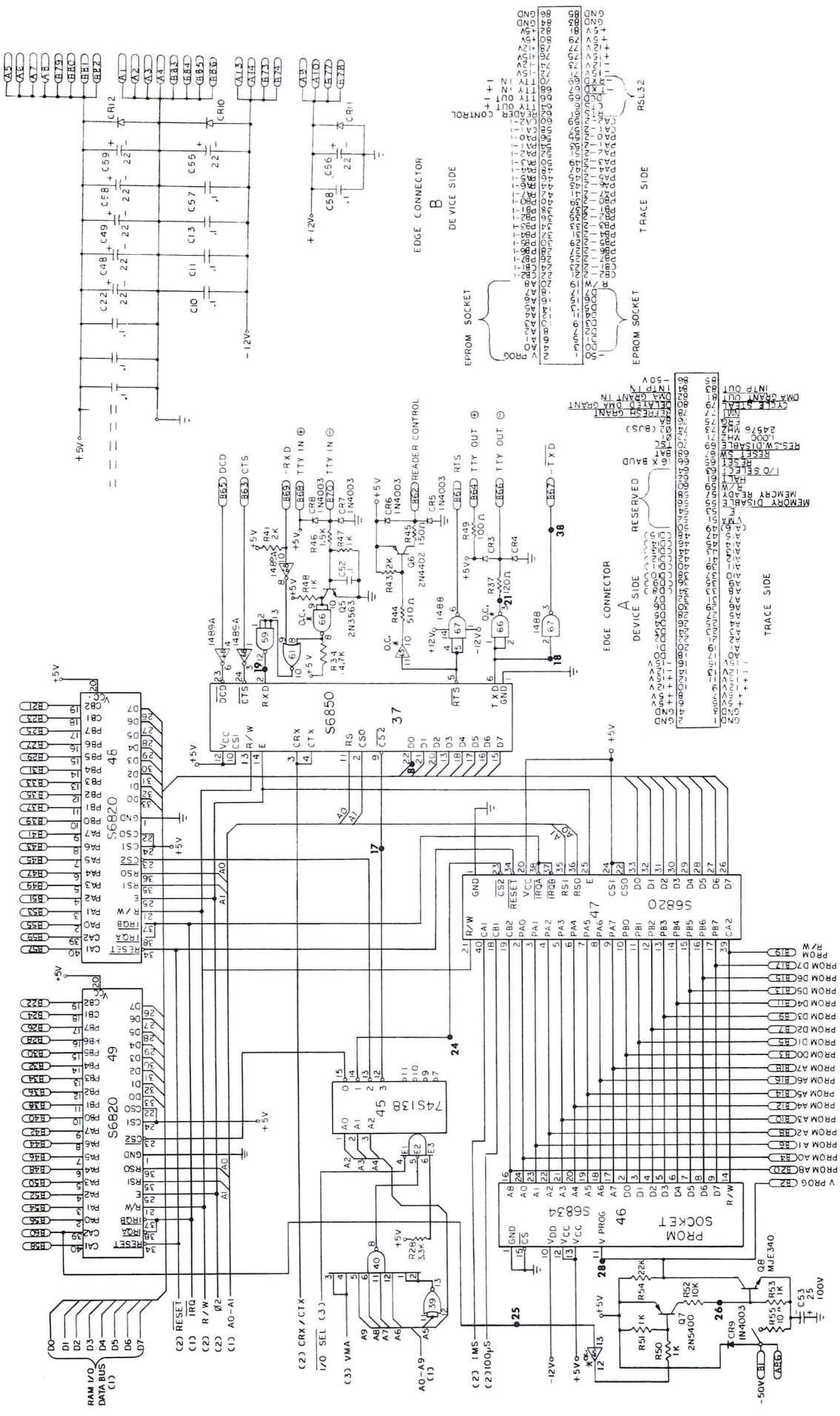


Figure 5. AMI Prototyping Board Schematic Diagram

EVK MICROCOMPUTER FUNCTIONAL DESCRIPTION

FUNCTIONAL CONFIGURATION

The following functional description is directed towards the fully populated EVK 300 Microcomputer board and is functionally applicable to the complete series of EVK boards limited only by the degree of board hardware-software population.

No attempt will be made to functionally describe the characteristics of AMI's Microcomputer IC chip set as this was undertaken in the first article entitled AMI 6800 Microcomputer Chip Set published last month in INTERFACE AGE. Instead, we will describe the general architecture of the EVK Microcomputer board and its general characteristics in order to provide an insight into EVK board utilization.

GENERAL ORGANIZATION

The EVK Microcomputer functional configuration is composed of the following major functional sections: MPU, clock, internal timer, memory, EPROM programmer, internal bus, expansion bus, I/O bus, I/O, and control logic sections. This functional interrelationship is shown in Fig. 1, EVK Microcomputer Functional Block Diagram while the detailed logic and circuit information is shown in Fig. 2, 3, 4, & 5, EVK Microcomputer Logic Diagrams.

MPU — The MPU is mechanized with AMI's S6800 Microprocessor chip. All MPU data address and control lines are buffered, and in addition are available at the board edge connector.

MPU TWO-PHASE CLOCK — The basic MPU two phase clock is derived from a 96S02 dual one-shot (IC12) connected either in a regenerative feedback loop or driven by a 1 MHz crystal controlled oscillator circuit (IC14). Switch #SW 2 is used to select either the one-shot regenerative feedback or the crystal oscillator mode of operation. Phase one and two timing is controlled by potentiometers connected to the one-shot RC timing networks and controls the phase pulse widths. These two phase additive pulse widths determine the MPU clock rate when the one-shot regenerative feedback configuration mode is connected. In this regenerative feedback mode, MPU clock frequency may be adjusted from 300 KHz up to 1 MHz. The phase timing outputs of the one-shots in both modes of operation drive 2N5771-2N5772 transistor amplifier circuits which in turn drive the two phase clocks of the MPU. In addition, both clock phases are buffered and available at the board edge connector. The fixed frequency 1 MHz crystal oscillator circuit output is also buffered and available at the board edge connector.

The two phase clock can be halted in either phase 1 or phase 2 for cycle-steal, DMA or slow memory applications. Phase 1 is halted (held HIGH) by driving the $\overline{\text{CYCLE STEAL}}$ control line LOW. Phase 2 is halted (held HIGH) by driving the $\overline{\text{MEMORY READY}}$ line low. Because the S6800 internal registers are dynamic and must be refreshed periodically $\overline{\text{CYCLE STEAL}}$ and $\overline{\text{MEMORY READY}}$ line outputs to the one-shots cannot be held LOW for more than 5 μs. This time limit protection, regardless of control input conditions, is provided by open collector 7407 (IC65) Hex non-inverting drivers, disconnect diodes and one-shot RC pull-up timing networks.

INTERNAL TIMER — A crystal controlled interval timer provides 100μs and 1 ms time periods for interrupting the MPU for real time clock applications. The 1 MHz crystal clock output drives a three decade divide-by-ten 74160 counters (IC50, 51, & 52) which in turn provide the 100μs and 1 ms time intervals. The 100 μs time interval pulse sets bit 7 of I/O address FBC7 via the S6820 PIA (IC47) while the 1 ms time interval pulse sets bit 7 of I/O address FBC5 via the S6820 PSA. These two time interval signals are used for timing EPROM programming.

MICROCOMPUTER BUS ARCHITECTURE — The EVK Microcomputer in essence has three sets of busses, namely the MPU bus, the Microcomputer bus and the on board memory-I/O bus. Each bus set consists of an 8-bit bidirectional data bus, a 16-bit unidirectional address bus and a control bus. The MPU bus is isolated from the Microcomputer bus in order to keep MPU signal loading to a minimum. The on board memory-I/O bus is isolated from the Microcomputer bus in order to assure that the on board memory and I/O devices do not load down the Microcomputer bus. As a result of this load isolation 40 ma drive current is available to drive external expansion hardware. The bus isolation buffers are non-inverting 3-state hex buffers (8T97). All of the controls to and from the S6800 are available at the board edge connector. This allows the user complete access and control of the MPU. Bus logic polarity is the same on all three busses (logic true = voltage high = "1"). The enable control signals to the MPU are always active. Control signals for the address bus are gated by the DMA GRANT line. The data bus is controlled by the DMA and R/W Lines.

MEMORY — The on board memory includes 1K bytes static RAM, 4K bytes ROM and 2K bytes EPROM.

MEMORY ADDRESS ASSIGNMENTS — Address assignments have been made such that all components on the card can run in the upper 8K bytes of memory. An address assignment map is shown in Figure 6.

Address decoding is made by use of three 74S138 one-of-eight decoders (IC 44, 45, 54). The first decoder (IC 54) selects one 1K-byte block of the upper eight 8K-bytes of memory. The output of this decoder is for RAM, I/O, ROM, or PROM enable lines. The second decoder (IC 44) selects one of eight RAM memory chips. The third (IC 45) selects I/O devices on the board.

A MEMORY DISABLE line is available at the Bus edge connector. This line, when LOW, deselects the first address decoder disabling all I/O and memory devices on the board. An I/O ENABLE line is derived from the first adress decoder and is available at the Bus edge connector. It must be noted that I/O ENABLE on the backplane is not valid when MEMORY DISABLE is LOW.

READ ONLY MEMORY — The Prototyping Board has assigned locations for two 1K byte S6830 ROMs and for four 512 x 8 S6834 EPROMs. The ROM circuits are designed such that the locations will also accept two 2K byte 16K ROMs (S6831). Thus, maximum memory allocation for ROM and EPROM is 6K bytes. The prototyping operating system program (PROTO) is assigned to the ROM with a starting address of F000.

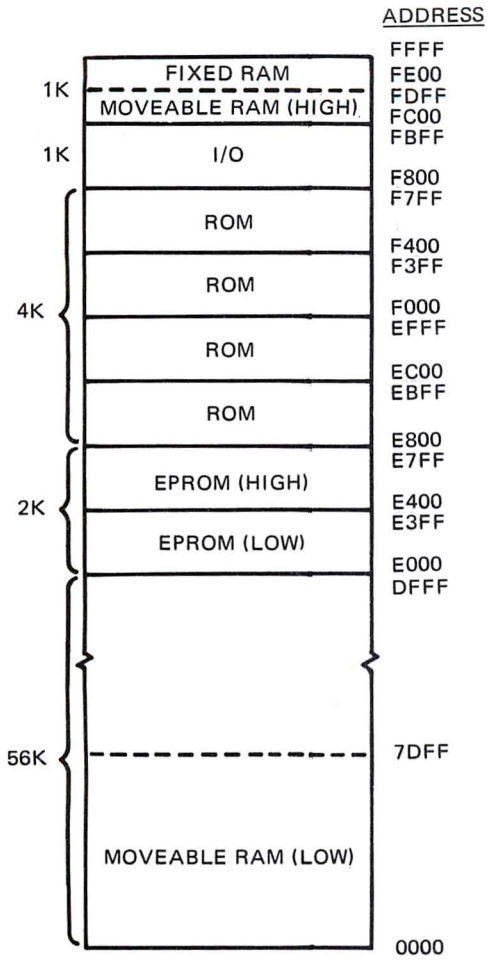


FIGURE 6. MEMORY ASSIGNMENT MAP FOR THE AMI PROTOTYPING BOARD

The four EPROM locations may contain any user program. Execution can start from beginning EPROM location either by selecting EPROM starting address of E000 in the restart switches or by branching to that address using the "G" command in the PROTO program.

RANDOM ACCESS MEMORY — The RAM is divided into two parts, 512 bytes fixed in the highest memory locations and 512 bytes of moveable memory.

Since the highest memory locations (FFFE, FFFF) are used for restart address, the address circuits disable the RAM using a memory disable line and force the 16 bit switch address on the data bus whenever a Reset occurs. This allows the user to vector to any address as his restart address.

The PROTO program assigns restart vectors for IRQ, NMI, and SWI whenever it is started (usually via Reset). It is therefore important to note that the user program must do the same thing if he does not use PROTO and restarts from a power down mode.

The stack pointer is assigned to address FF8F in PROTO. This allows the remaining RAM to be used as stack if so desired.

A switch option allows 512 bytes of RAM to be relocatable. When in the upper portion of memory, the RAM is assigned to addresses FC00 to FDFF making all 1K-bytes of RAM on the board contiguous (FC00 to FFFF). When in the lower portion of memory, the 512 bytes are addressed whenever A9 and A15 are not true (0000 — 01FF for example). It is thus recommended that RAM be assigned to the low address only if the user does not add other RAM to his development system.

I/O — On board I/O includes parallel PIA I/O ports and serial ACIA TTY and RS232C I/O ports.

PARALLEL I/O— Three S6820 PIA's give the user a wide range of I/O flexibility. The PIA's are assigned addresses as shown in Table 2. Interface pins of these devices are directly connected to the I/O edge connector. The CA2 pin for the PIA at addresses FBC4 is also connected to the VPROG input (pin 11) to the EPROM socket (IC 46) through a +5V to −50V driver. The user is cautioned to use this line such that it will not interfere with his I/O function if programming an EPROM. For example, if the CA2 line is connected to an external control function, this function may be erroneously activated while programming an EPROM.

TABLE 2. I/O ADDRESS ASSIGNMENT

I/O PORT	ADDRESS	ASSIGNMENT
S6850 ACIA		Serial I/O – TTY
	FBCE	Status/Read
	FBCF	Control/Write
S6820 PIA 1		Unassigned
	FBC8	Peripheral Register A
	FBC9	Control Register A
	FBCA	Peripheral Register B
	FBCB	Control Register B
S6820 PIA 2		Keyboard/Unassigned
	FBC0	Peripheral Register A
	FBC1	Control Register A
	FBC2	Peripheral Register B
	FBC3	Control Register B
S6830 PIA 3		PROM Burner
	FBC4	Peripheral Register A
	FBC5	Control Register A
	FBC6	Peripheral Register B
	FBC7	Control Register B

SERIAL I/O — One S6850 ACIA allows the system to communicate bi-directionally with serial data I/O peripherals such as a TTY. A baud rate generator generates all standard communication frequencies by switch selection. This frequency operates independently of the system clock so the MPU frequency can be changed without altering the I/O clock rate. See Table 3 for switch setting and associated frequencies. A 20 mA current loop interface and an RS-232 interface are both available at the I/O edge connector.

Address assignments for the ACIA are given in Table 3, "Bit Rate Generator Switch Settings."

EPROM PROGRAMMER — A unique feature of the Prototyping Board is its ability to program AMI S6834 EPROMs. A third PIA latches the address and data information for programming the EPROM. The EPROM socket programs only the S6834 EPROM, however, an adapter plug is available to also program the AMI S5204A EPROM. Except for the VPROG input, all address, chip select, R/W and data I/O pins on both EPROMs are completely TTL compatible and are driven directly from the PIA outputs. The outputs are also available on the I/O edge connector for convenience in using another EPROM programming socket.

TABLE 3. BIT RATE GENERATOR SWITCH SETTINGS,

0 = CLOSED, 1 = OPEN

SW POSITION				BIT RATE
4	3	2	1	
0	0	0	0	19,200 baud
0	0	0	1	0 baud
0	0	1	0	50 baud
0	0	1	1	75 baud
0	1	0	0	134.5 baud
0	1	0	1	200 baud
0	1	1	0	600 baud
0	1	1	1	2,400 baud
1	0	0	0	9,600 baud
1	0	0	1	4,800 baud
1	0	1	0	1,800 baud
1	0	1	1	1,200 baud
1	1	0	0	2,400 baud
1	1	0	1	300 baud
1	1	1	0	150 baud
1	1	1	1	110 baud

Programming is achieved by pulsing the VPROG pin with −50 volts through the CA2 line of the PIA at address FBC4. This line drives the transistor that gates the −50 volt source to the VPROG pin. The −50 volt source is switched ON or OFF via the VPROG switch.

CONTROL — The Microcomputer control section includes system reset logic, addressable reset logic and DMA control logic. In addition, an external logic circuit may be added to provide selection between RUN and single step modes.

RESET — The Reset circuit provides a timed reset for Power On Reset timing and for the Reset switch. The circuit is a timed oscillator which provides a 200 ms reset pulse.

RESTART — The starting address of an S6800 is FFFE/FFFF. The contents of these memory locations are put into the Program Counter register each time the MPU is reset. The Evaluation Board traps the FFE/FFFF addresses and puts the contents of the two 8-bit switch sets (IC 32, 43) on the data bus for each address and disabling memory, then gating the first set of switches to the Data Bus during FFFE time and the second set during FFFF time. The user is thus allowed to select any restart address by simply selecting a two byte address on the 16 bits of switch settings. The two DIP switches may be replaced with four hex thumbwheel switches mounted on a front panel and interconnected via a flat ribbon cable and DIP plug connectors providing front panel Hex restart control.

DMA — Three types of DAM implementation are possible on the Prototyping Board, a halt processor mode, a cycle steal mode and a multiplex mode. A switch selects these DMA modes. The switch must be in the DMA position for the multiplex DMA mode. A delayed clock gives the DMA GRANT line to the bus after the "Data Hold" time has passed for a multiplexed type of DMA operation. The control lines for the halt processor and cycle steal modes are available at the Bus edge connector.

RUN/HALT & SINGLE STEP EMBELLISHMENTS — A simple low cost three IC RUN/HALT-Single Step Instruction logic may be added external to the EVK board to provide these capabilities if required. Figure 7, RUN/HALT & Single Cycle Instruction Logic Diagram and Figure 8, Single Step Timing Diagram depicts this added logic mode.

POWER REQUIREMENTS

The EVK board is mechanized so that nominally only a +5 volt @ 3.5 amp power supply is required. A −12 volt supply is required when using S6834 EPROM ICs. In addition, a −50 supply is required when programming these EPROMs. The RS232C Interface requires both the +12V and −12V supplies for proper operation. The following is the total power and voltage level requirement for a complete operational *EVK 300* Microcomputer board;

+5V @ 4 Amps
−12V @ 150 ma
+12V @ 50 ma
−50V @ 50 ma

SOFTWARE

The EVK 300 Prototyping Board Software is comprised of a TTY Operating Program (PROTO) and is supported by a ROM Subroutine Library (RS)[3].

PROTO— The EVK 300 is supplied with a prototyping operating system program (PROTO). The program resides in ROM with a starting address of FØØØ. The various routines within PROTO are called by entering via the TTY keyboard one of the commands. A command consists of one character command identifier followed by additional parameters, if needed, separated by blanks or commas. All commands end with a carriage return. Since no action is taken before the carriage return, an input line may be deleted by the use of the TTY ESCAPE key. The PROTO program operates on the following commands:

L Load Memory from TTY paper tape (HEX Format)

P Punch a Memory location to TTY paper tape (HEX Format)

S Set (write) Memory to a given value

D Display the contents of a memory location in HEX

G Go to user program at specific address and begin program execution

R Print contents of MPU C, B, A, X, P & S register on the TTY

B Burn (program) an EPROM from Memory location indicated

V Verify the contents of an EPROM with a specified memory location

I Input (copies) contents of EPROM in the programming socket into memory.

M Move a specific block of memory to a designated location

E End of transmission (EOT) character terminates the record and punches EOT on paper tape.

The commands will operate on a single character op code plus address parameters from the TTY keyboard.

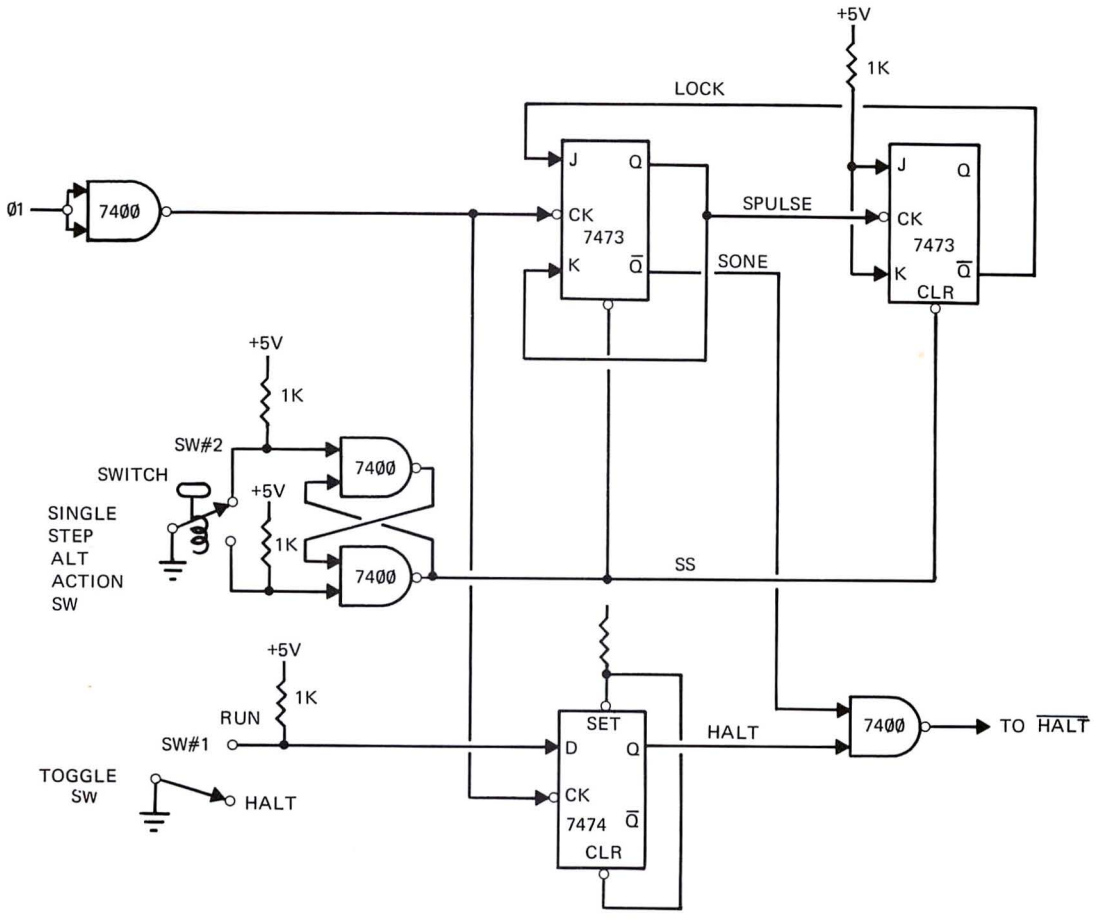


Figure 7. Run/Halt and Single Cycle Instruction Logic Diagram

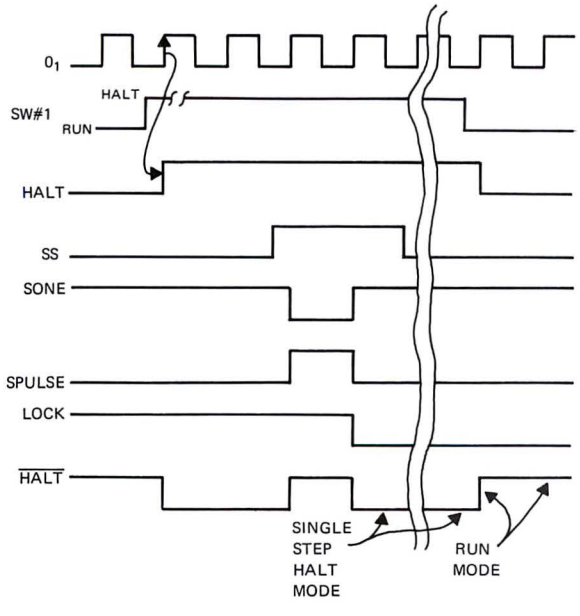


Figure 8. Single Step Timing Diagram

(RS)³ SUBROUTINES — The 2K X 8 ROM provided with the PROTO prototyping system includes a set of (RS)³ subroutines with a slightly different linkage from the standard (RS)³ form, although the calling sequence is the same. In particular, the provision for additional subroutines in the of other (RS)³ ROMs is limited to a total of 127 subroutines. The first additional (RS)³ ROM address must be placed in RAM location FFF4 (which can be set via the Set Memory command or modified by an initialization code in a user program). Also, since it is incorporated into a larger program, the whole of which very nearly fills the 2K bytes of its ROM, the (RS)³ part of the ROM does not start on an even page boundary, making it awkward for isolated use. However, the 24 subroutines included in this ROM are available to user program calls with the SWI calling sequence, as described.

The ROM Subroutine Library (RS)³ operates on a single SWI (3F) command and a second byte of offset giving the S6800 an additional set of two-byte instructions. Specific subroutines (offsets) are as follows.

SUBROUTINE INDEX	MNEMONIC	FUNCTION
0	PUSHALL	All registers are pushed on to user stack.
1	POPALL	All registers on user stack are loaded into MPU.
2	TXAB	Contents of Index Register are transferred to A & B Accumulators.
3	TABX	Contents of A & B Accumulators are transferred to Index Register.
4	XABX	Contents of A & B Accumulators are exchanged with contents of Index Register.
5	PUSX	Contents of Index Register are pushed onto user stack.
6	PULX	Index Register is loaded with contents of user stack.
7	ADDXAB	Contents of Index Register are added to contents of A & B Accumulators. Sum is in A & B Accumulators.
8	ADDABX	Contents of A & B Accumulators are added to contents of Index Register. Sum is in Index Register.
9	ADDAX	Contents of Accumulator A are added to contents of Index Register. Sum is in Index Register.
10	ADDBX	Contents of Accumulator A are added to contents of Index Register. Sum is in Index Register.
11	SUBXAB	Contents of Index Register are subtracted from contents of A & B Accumulators. Remainder is in Accumulators A & B.
12	SUBABX	Contents of Accumulators are subtracted from contents of Index Register. Remainder is in Index Register.
13	SUBAX	Contents of Accumulator A are subtracted from contents of Index Register. Remainder is in Index Register.
14	SUBBX	Contents of Accumulator B are subtracted from contents of Index Register. Remainder is in Index REgister.
15	P2HEX	Two Hexidecimal Characters (one MPU byte) are printed on the TTY.
16	P4HEX	Four Hexidecimal Characters (two MPU bytes) are printed on the TTY.
17	PRINTA	ASCII Character designated is printed on TTY.
18	PMESS	Message designated is printed on TTY.
19	VALAN	Character (byte) is checked to see if it is a valid alpha/numeric character.
20	INPUTA	ASCII Character at TTY is input to MPU.
21	CONHB	ASCII Character string is scanned looking for a valid Hexidecimal number.
		Binary equivalent is returned in Accumulators A & B.
22	INDEX	Contents of Accumulator A are multiplied with the contents of Accumulator B and the product is added to the contents of the Index Register.
23	MUL8	Contents of Accumulator A are multiplied with the contents of Accumulator B. Product remains in both Accumulators.

S6800 MICRO ASSEMBLER/DISASSEMBLER (MA/D) — An optional ROM resident Micro Assembler/Disassembler is available for the EVK Microcomputer board at an additional cost of $30.00. Where this option is provided for those applications it may be desirable to debug programs using the mnemonic instruction codes instead of hexadecimal values. MA/D is designed to accomplish this by interfacing with a user via a keyboard and display (TTY or equivalent). The required 6800 environment must include:

Character in routine at location	0
Character out routine at location	3
No. nulls after carriage return at location	6
RAM at locations	7 - 78 $_{10}$

The I/O routines must transfer the characters in Register A and return with a RTS. It is expected that location 0 will just include a JMP to the actual character in routine, or, in the case of AMI's proto board:

```
00 SWI
01   FCB 20
02 RTS
```

The stack pointer must also be initialized before MA/D is entered. MA/D itself can execute from ROM, located anywhere in the system. MA/D may be started at its beginning address +2, in which case it will set up its environment for the AMI proto board.

After entering MA/D, the line length may be changed. The line length is in location 7 and is initially set to $(20)_{10}=14$ hex. The line buffer itself begins in location $(58)_{10}=3A$ hex.

After displaying a header message MA/D prompts the user for a command by displaying MA/D's current location counter followed by a colon (:). The commands available to the user allow for disassembly of instructions in memory and assembly (mnemonic translation and operand insertion with relative offset computation) of instructions directly into memory.

MA/D is also very useful for writing short test programs. The instruction format for assembly is identical to the S6800 Assembler except:

1) operands must be in hexadecimal without the $, and no more than four digits long
2) no symbols can be defined or referenced
3) relative addresses are specified as absolute addresses, the offset is computed
4) in those instructions having **both** direct and extended addressing modes, extended addresses must have at least three digits. Thus,
   LDA A 10 assemblies as 96 10
   LDA A 010 assemblies as B6 00 10
5) in those instructions not having a direct addressing mode, the operand may be two or more digits. Thus,
   INC 10 assembles as 7C 00 10
6) an operand may be a single hex digit only if the op code indicates an A or B register, or immediate mode addressing. Thus,

—are legal—	—are not—
INC 01	INC 1
LDA A 1	LDX 1
LDX #1	

   (This makes it easier to distinguish between, for instance, INC A and INC 0A.)
7) Anywhere a number is used, the construction 'character may be used instead, and is equivalent to the ASCII code for the character.

MA/D Command Summary

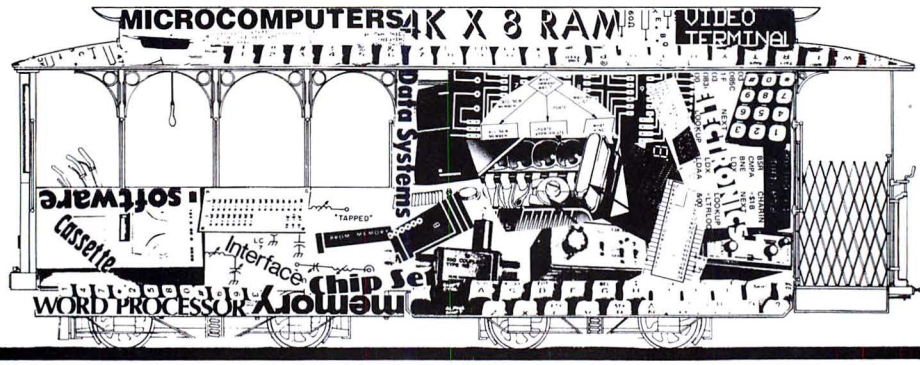
@newloc @ newloc	The @ sign followed (immediately or with blank separator) by a hexadecimal address initializes the current location counter to the new address. MA/D automatically updates the location counter as instructions are assembled or disassembled.
$count $ count	The $ sign followed by a one or two digit (hexadecimal) count results in the disassembly of "count" instructions. Zero = infinity.
!address ! address	The exclamation mark followed by an address causes MA/D to call a subroutine at the given address. If the subroutine returns with the carry flag set, MA/D will print "????".
!	Exclamation mark with no address given causes MA/D to call the subroutine starting at the current location.
"string	Assembles the ASCII characters following the double quote mark into successive bytes of memory starting at the current location. The current location is updated.
xx x 'character	A one or two digit hexadecimal number is placed into the current location, and the current location is incremented by one. A single quote mark followed by a single character causes the ASCII code for that character to be placed in the current location. This command may appear several times on the same line, the numbers or quoted characters separated by spaces or commas.
&address,count & address,count &address count & address count	Ampersand followed by a hexadecimal address and count (from 1 to 4 digits each) causes "count" bytes to be moved from "address" to the current location. On completion, the current location is incremented by "count".
<RETURN>	Carriage return is equivalent to $01, disassemble a single instruction.

The commands to MA/D are buffered and not processed until the ‹RETURN› key is depressed. The ‹BACKSPACE› key can be used to delete the last character input. If errors are detected on user input the line is ignored, ???? is displayed, and another prompt is issued.

The default command is "assemble" and MA/D, if not recognizing the input as one of the following commands, generates the machine code for the instruction mnemonic.

Next month we will publish the complete PROTO Assembly Listing for the EVK Microcomputer board.

```
>
>G E002
A.M.I. 6800 MICRO ASSEMBLER/DISASSEMBLER — 1.0
(C) 1976, A.M.I.
002A:@80
0080:"THIS IS LOOP NO.
0090:"0000
0094:04
0095:LDA A 93
0097:INC A
0098:STA A 93
009A:CMP A #3A
009C:BNE 110
009E:LDA A '0
00A0:@9E
009E:
009E-> 96 LDA A 30
00A0:@9E
009E:LDA A #'0
00A0:STA A 93
00A2:LDA A 92
00A4:INC A
00A5:STA A 92
00A7:BRA 110
00A9:@110
0110:LDX #0080
0113:LDA A 00,X
0115:CMP A #04
0117:BEQ 120
0119:JSR E003
011C:@119
0119:JSR 0003
011C:INX
011D:BRA 113
011F:NOP
0120:LDA A #0D
0122:JSR 0003
0125:LDA A $0A????
0125:LDA A #0A
0127:JSR 0003
012A:JMP 095
012D:@95
0095:$3
0095-> 96 LDA A 93
0097-> 4C INC A
0098-> 97 STA A 93
009A:!95THIS IS LOOP NO.0001
THIS IS LOOP NO.0002
THIS IS LOOP NO.0003
THIS IS LOOP NO.0004
THIS IS LOOP NO.0005
THIS IS LOOP NO.0006
THIS IS LOOP NO.000
>
>
```



THE FIRST WEST COAST COMPUTER FAIRE

A Conference & Exposition on Personal & Home Computers

San Francisco Bay Area—where it all started—finally gets its act together.

7,000-10,000 PEOPLE ▪ 100 CONFERENCE SESSIONS ▪ 200 COMMERCIAL & HOMEBREW EXHIBITS

2 BANQUETS ▪ SPECIAL INTEREST SOCIAL CENTERS

San Francisco Civic Auditorium, Northern California's Largest Convention Facility

YOU Can Be A Part Of It:

▪ *Some* of the things *you* can do are:

Exhibit a Homebrewed System
- Hardware or Software
- Prizes for Best "Homecooking" (just like a country fair)
- Nominal Grants-In-Aid Will Help With Exhibit Transportation Costs (grants will be refereed)

Present a Talk
- A Formal Paper
- An Informal Talk

Serve on a Panel
- As a Panel Member
- As the Coordinator/Moderator

Offer Suggestions
- Topics for Talks & Panels
- Speakers & Panel Participants
- Interesting Exhibits (homebrewed or commercial)
- Special Activities

▪ *Quick*, write or call for more details!
Jim Warren, Faire Chairperson
Box 1579
Palo Alto CA 94302
(415) 851-7664 v 323-3111

▪ *Some* of the **Conference Sections** being planned:
- Computer Graphics on Home Computers
- Computer-Driven, & Computer-Assisted Music Systems
- Speech Synthesis Using Home Computers
- Computers & Amateur Radio
- Microprogrammable Microprocessors for Hobbyists
- Program & Data Input via Optical Scanning
- Floppy Disc Systems for Personal Computers
- Computer Games: Alphanumeric & Graphic
- Computers & Systems for Very Small Businesses
- Personal Computers for the Physically Handicapped
- Personal Word-Processing Systems
- Software Design: Modularity & Portability
- Personal Computers for Education associated with a Univ. of California short-course
- Several Sections Concerning Standards
- Other Sections for Club Leaders, Editors, Organizers, etc.

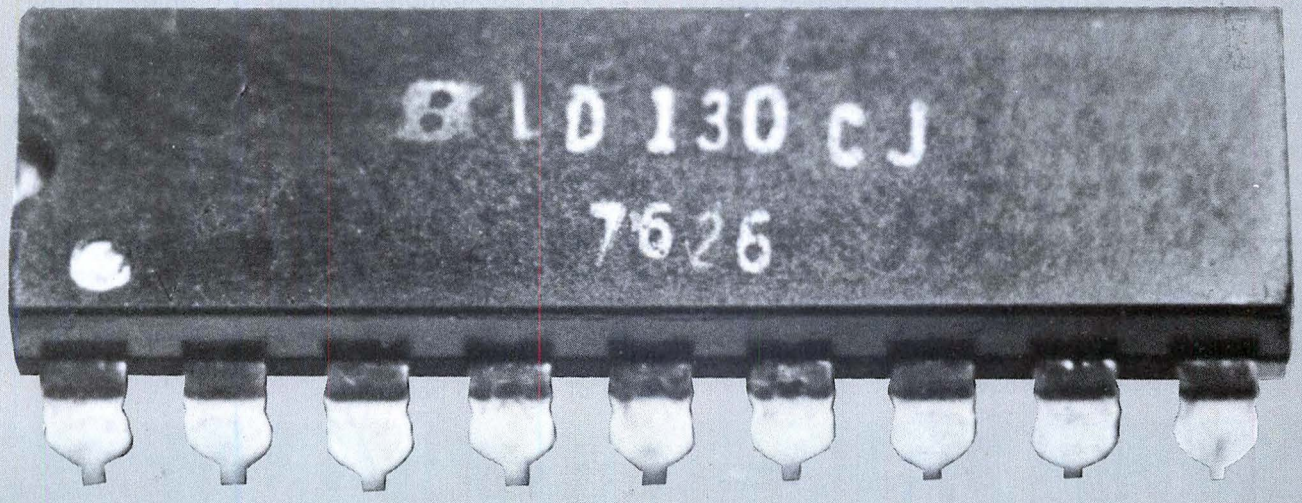
▪ **Co-Sponsors** include amateur, professional, & educational groups:
- The Two Largest, Amateur Computer Organizations
  Homebrew Computer Club
  Southern California Computer Society
- Both Area Chapters of the Association for Computing Machinery
  San Francisco Peninsula Chapter & Golden Gate Chapter
- California Mathematics Council
- Stanford University's Electrical Engineering Department
- Community Computer Center
- People's Computer Company

april 15-17, 1977 · san francisco

CIRCLE INQUIRY NO. 41

Hardware Report:

Build a 3 Digit A/D Converter for Your Microcomputer



By Roger Edelson

For the next two months Hardware Report will cover the Siliconix LD130 Monolithic CMOS A/D Convertor. This Device is admirably suited to provide a general purpose analog-to-digital (A/D) interface for your microprocessor system that requires only one chip (the LD130), and a minimum of additional components. This device will provide a 3-digit bi-polarity (both positive and negative — effectively doubling the conversion range) A/D conversion at rates up to 60 samples per second. Using a two transistor decode scheme, which will be shown later, the inputs to the LD130 can be multiplexed to provide a number of channels of A/D conversion. However, using an input multiplexing scheme reduces the rate of conversions for any single channel by the number of channels multiplexed. In the case of a four paddle type game the maximum conversion rate for any one channel is, therefore, 60 divided by 4, or 15 conversions per second. This rate may be not fast enough in which case multiple LD130s' could be used or, alternatively, a simpler A/D scheme, like a strobed one-shot could be used, as a 3-digit accuracy would not be required. Bearing this one limitation in mind, the LD130 is, however, ideal where 3-digit accuracy is required, and simplicity of hook up is desirable.

The LD130 combines both the analog and digital sub-systems of a 3-digit A/D system in a single monolithic CMOS I.C. The 'Quantized Feedback' conversion scheme, introduced by Siliconix, provides the LD130 with an Auto-Zero, Auto-Polarity A/D system requiring only a single reference voltage. External parts are minimized by the on-chip resistors and buffer amplifiers. These high impedence input and reference buffer amplifiers eliminate source loading errors providing the outstanding temperature coefficient and ratio operation inherent in this system. Break-before-make switching action insures that neither the analog input nor the reference voltage will be shorted to ground at any time.

Figure 1 shows the internal functional arrangement of the Siliconix LD130, as well as the connections of the external components. The pin configuration of the dual-in-line package is also shown.

Let's look at some of the features of this ±3-digit A/D converter.

1. Accuracy of 0.1% (of reading, not full scale) ±1 count.
2. Auto-Zero (utilizes break-before-make switch action to insure that the input signal is not shorted to ground).
3. Auto-Polarity.
4. Low power consumption (25mw typical).
5. 1mv resolution (1.00V full scale).
6. Minimum of external components:
   a. C_{AZ} (0.10μf) between AZ and Σ pins.
   b. C_{INT} (0.033μf) between INT and Σ pins.
   c. C_{OSC} (0.001μf) between OSC and digital ground.
   d. $V_{REF} \cong$ 2.00 V. must be provided.
   e. ±5V supplies (@ 3ma) are needed.
7. Buffered Analog Input ($Z_{IN} >$ 1000MΩ)
8. Buffered Reference Input ($Z_{IN} >$ 1000 MΩ)
9. Internal Clock Oscillator (adjustable by selection of C_{OSC} or an external clock may be used.)
10. Overrange and Underrange Signals provided for Auto-Ranging.
11. Multiplexed BCD with inter-digit blanking (to minimize interface connections).
12. TTL Compatible outputs (1 load driving capability).
13. Sampling Rates from 1 to 60 Samples per second.

Table 1 gives the specifications of the LD130 chip. Note that many of the error sources are given as typical only. The typical values are given as a design aid ***only***, and are neither guaranteed nor subject to production testing. In most cases your chip will conform to these values, and unless you really need to work close to the maximum error budget for some applications, they will not affect your usage of the chip. Let's discuss some of the chip parameters:

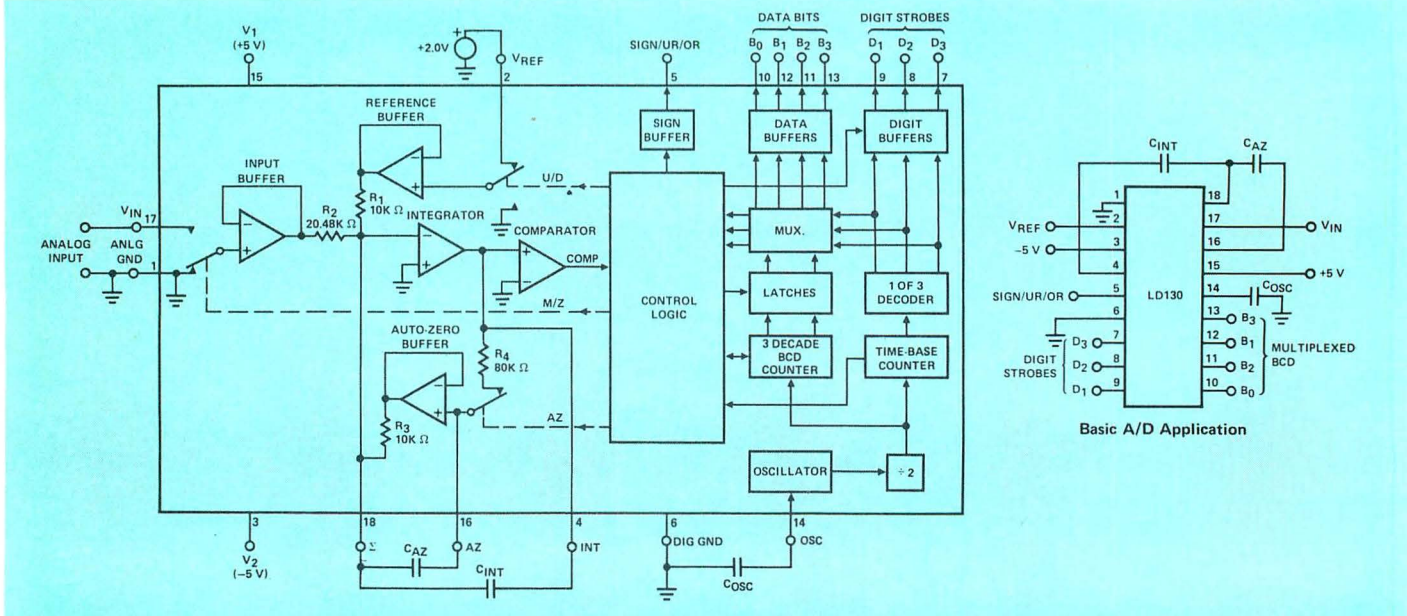
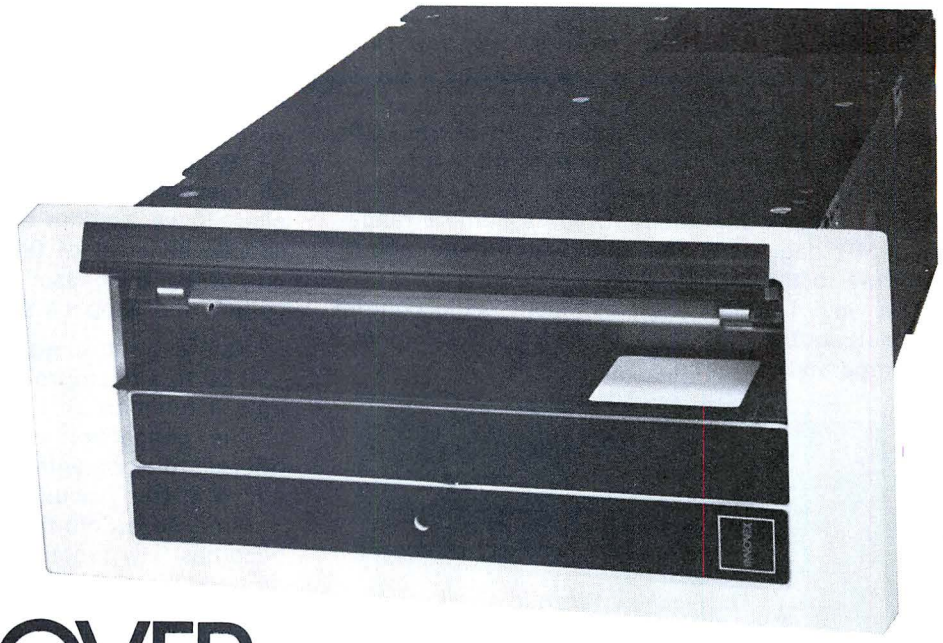


Figure 1. LD 130 — Functional Diagram



FLIP OVER OUR FLOPPY

Only $750 from Peripheral Vision.

Peripheral Vision is a brand-new company that's dedicated to selling reasonably priced peripherals for various manufacturers' CPU's.

We think you'll flip over our first product.

It's a full-size floppy disk for the Altair-Imsai plug-in compatible S-100 BUS. And it's available for as low as $750.

Here are the features:

- 1 interface card supports 4 drives
- Stores over 300,000 bytes per floppy
- Bootstrap EPROM included—no more toggling or paper tape
- Completely S-100 plug-in compatible
- Interface cabling included
- Drive is from Innovex (the originator of the floppy concept)—assembled and tested
- Interface card design is licensed from Dr. Kenneth Welles and the Digital Group
- Disk operating system with file management system included on floppy
- Cabinet and power supply optional

Prices:	Kit	Assm.
Interface card kit and assembled and tested drive	$750	$850
Power supply—+24V at 2A	45	65
Cabinet—Optima, blue	—	85

Now, a little more about our company.

Peripheral Vision may be brand-new, but we have some old-fashioned ideas about how to run our business.

We know there are serious incompatibilities among the different manufacturers' peripherals and CPU's. We want to get them together. And, we want to bring significant new products to market—products consisting of everything from adaptation instructions/kits for hardware and software to major new products.

It's a tall order, but we feel we're up to the task. Peripheral Vision has already obtained a license from The Digital Group to adapt versions of some of their products to the S-100 BUS. And we're working on getting more from other companies.

Most important to our customers, Peripheral Vision is committed to helping you get along with your computer. We'll do all we can to make it easy.

Write us now for all the information on our company, our philosophy and our exciting line of products. And be prepared to flip over all of it.

PERIPHERAL VISION

P.O. Box 6267 / Denver, Colorado 80206 / (303) 733-1678

Send me the works, and I just might flip over it!

Name_____

Address_____

City/State/Zip_____

1. Linearity is given as a maximum error of 0.1% of the reading. That is quite a bit better than 0.1% of full scale which could give you an error of 1mv on a 1mv input signal (1.000v full scale) which would be a 100% error.

2. Noise is typically given as one third of the LSB (least significant bit) or roughly one count error.

3. The Zero temperature coeffecient of zero setting is 10μv per degree/centigrade. If you assume a range of about 40 degrees/C this could contribute about ±0.200 mv to the error budget.

4. The Gain Temperature Coefficient could contribute about another ±0.03% error for the assumed 40°C temperature change.

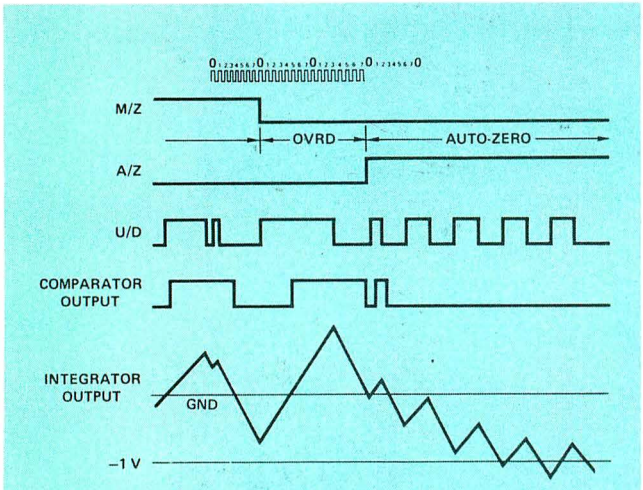


Figure 2. Auto-Zero Timing

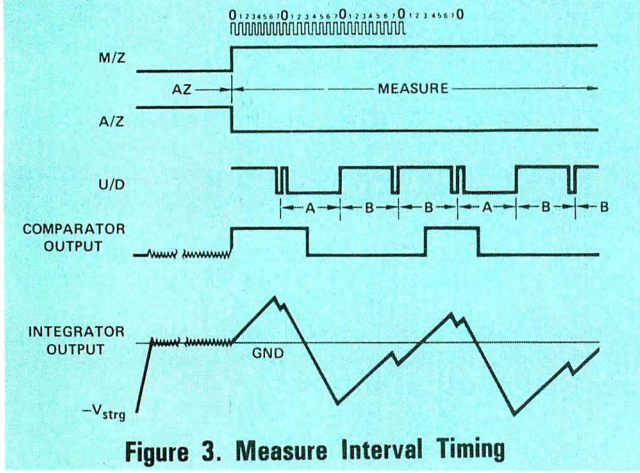


Figure 3. Measure Interval Timing

5. The normal mode noise rejection (NMR) is a minimum of 36db for a 60Hz noise frequency and a 24KHz clock. This gives a reduction of about 63 in the input noise. Later we will discuss a method for tuning the noise rejection of the LD130 to allow a maximum of over 70db noise reduction. This would give a reduction of over 3000 in the effect of the noise voltage.

6-9. These parameters define the input current for both the analog input and the reference input pins. At 70°C the input current will be less than 100pa which translates to >1000 megohms input impedence.

10-11. These two parameters define the clock conditions. Typically the clock frequency will be 30KHz with those components. The maximum clock frequency should be limited to ~400KHz. The duty cycle should be 50% ± 20%.

12-16. These items define the input and output parameters of the various logic and clock inputs. Note that the clock input requires 1ma of current in the high state. This is different than a standard TTL load.

17-18. Maximum limits for supply current. This device being a CMOS I.C. requires very little power.

19-20. These two parameters indicate the amount of rejection of noise contributed by the power supplies. If we assume a 200mv noise voltage on each supply, an error of 0.4mv will be contributed to the output. For this reason, in order to obtain maximum accuracy, the supplies should be fairly clean.

Before we discuss the application of the Siliconix LD130 to a microprocessor system let's take a look at how it functions.

The connection diagram of Figure 1 should be referred to along with the timing diagrams Figures 2, 3, and 4 in this discussion of functional operation.

Time Base Counter — The internal oscillator circuit becomes fully functional with an external capacitor to ground. The OSC input can be driven by an external oscillator (0 to V_I logic levels) if desired. A squaring circuit divides the oscillator frequency by two before it drives the BCD counter and Time-Base counter.

The two fundamental intervals of the sampling period, the Auto-Zero (AZ) and the Measure intervals, are established by the Time-Base counter as 1024 and 2048 clock periods respectively. The total sampling interval is then 3072 clock periods long. Since the internal clock is onehalf of the oscillator frequency, the sampling period is then 6144 (2x3072) oscillator periods. The Time-Base counter also divides the internal clock by eight. This division provides sets of eight clock periods (octets) which are used by both the data multiplexer as digit "ON" times and the control logic as U/D (Up/Down logic) duty cycle periods.

The Auto-Zero interval provides a means to null out the offset voltages of the amplifiers used in the LD130. In addition, it automatically establishes a second tracking reference voltage necessary for bi-polar A/D conversion.

The Auto-Zero sequence is initiated when the M/Z (Measure/Zero) signal switches the input buffer amplifier to analog ground. After a brief count-correcting override period, the AZ switch is closed connecting the AZ amplifier and Integrator together in a closed loop second-order system. During this time the control logic ignores the comparator output and pulses the U/D switch at a 50% duty cycle of 4 clock periods "Up" and 4 "Down" (See Figure 2).

If, for the moment, we ignore any error sources equalibrium of this closed-loop system is obtained when the average currents through R_1 and R_3 are equal and opposite. This is achieved when the V_{AZ}, the Auto-Zero voltage, is equal to $-\frac{1}{2} V_{REF}$ since $R_1 = R_3$. Establishing V_{AZ} and storing it on C_{AZ} gives the U/D logic the capability of switching either a + or − reference current to the Integrator during conversion. Thus when U/D is "Up" the U/D switch is connected to ground, and the current to the Integrator is

$$I_1 - I_3 = \frac{0}{R_1} - \frac{V_{REF/2}}{R_1} = -\frac{V_{REF}}{2R_1}$$

The POLY 88 Microcomputer System

If you are into computers or considering a system, the POLY 88 is the machine to contemplate.

HARDWARE

• Popular 8080 central processor • Single-board CPU with ROM, RAM, vectored interrupt, real time clock, single-step logic and serial I/O • Video interface card – generates video to TV monitor and provides parallel keyboard input port • Serial and cassette mini-cards plug directly into CPU with ribbon cables • Cassette • ROM monitor with powerful debugger, video software, bootstrap loader • Backplane and power supply on one board simplifies construction • Rugged 6 amp power supply • All circuit boards are high quality, double-sided with plated-through holes • System is compatible with a wide range of Altair peripherals • Minimum point to point wiring means that the POLY 88 kit can go together in three evenings!

ABOUT SOFTWARE

Software is the reason the POLY 88 was designed. The operator can proceed from higher level languages like BASIC to developing machine code with the aid of our assembler. Our BASIC is a full 8K BASIC with character and byte manipulation. Best of all, the programmer is finally free of the teletype emulation mode so the video display can be used to full advantage.

The video display provides a unique opportunity to write new types of programs and games. Characters (16 lines of 64) and graphics (48 by 128 grid) are part of the processor's memory, so the display may be altered rapidly — the entire screen written in less than 20 milliseconds.

POLY 88 hardware provides many additional features that programmers have come to expect, such as vectored interrupt and real time clock.

See it at your local computer store.

ARE YOU NEW TO COMPUTERS?

The POLY 88 was designed for ease of use. No one should have to learn all the inner workings of computers just to enjoy one at a reasonable price. With the POLY 88, you can "play" pre-developed programs or explore the world of computer languages as your interests expand.

THE POLY 88 IS FOR EVERYONE

Want to develop a new computer language? Want to fight Klingons? The POLY 88 provides a firm foundation upon which to build your interests and develop your skills.

POLY 88 SYSTEM PRICES:

SYSTEM 1 — Kit includes 8080 vectored interrupt processor with real time clock, ½ K of RAM and 1K monitor on ROM: Video Terminal Interface displays 16 lines of 32 characters on a video screen and has a keyboard input port; cabinet, backplane, and power supply; complete assembly, operation and theory manual. $595.

SYSTEM 2 — System 1 plus 64 character line option and Byte/biphase cassette interface kit. $690.

SYSTEM 3 — System 2 plus 8K of RAM with BASIC and assembler programs on cassette tape. $990.

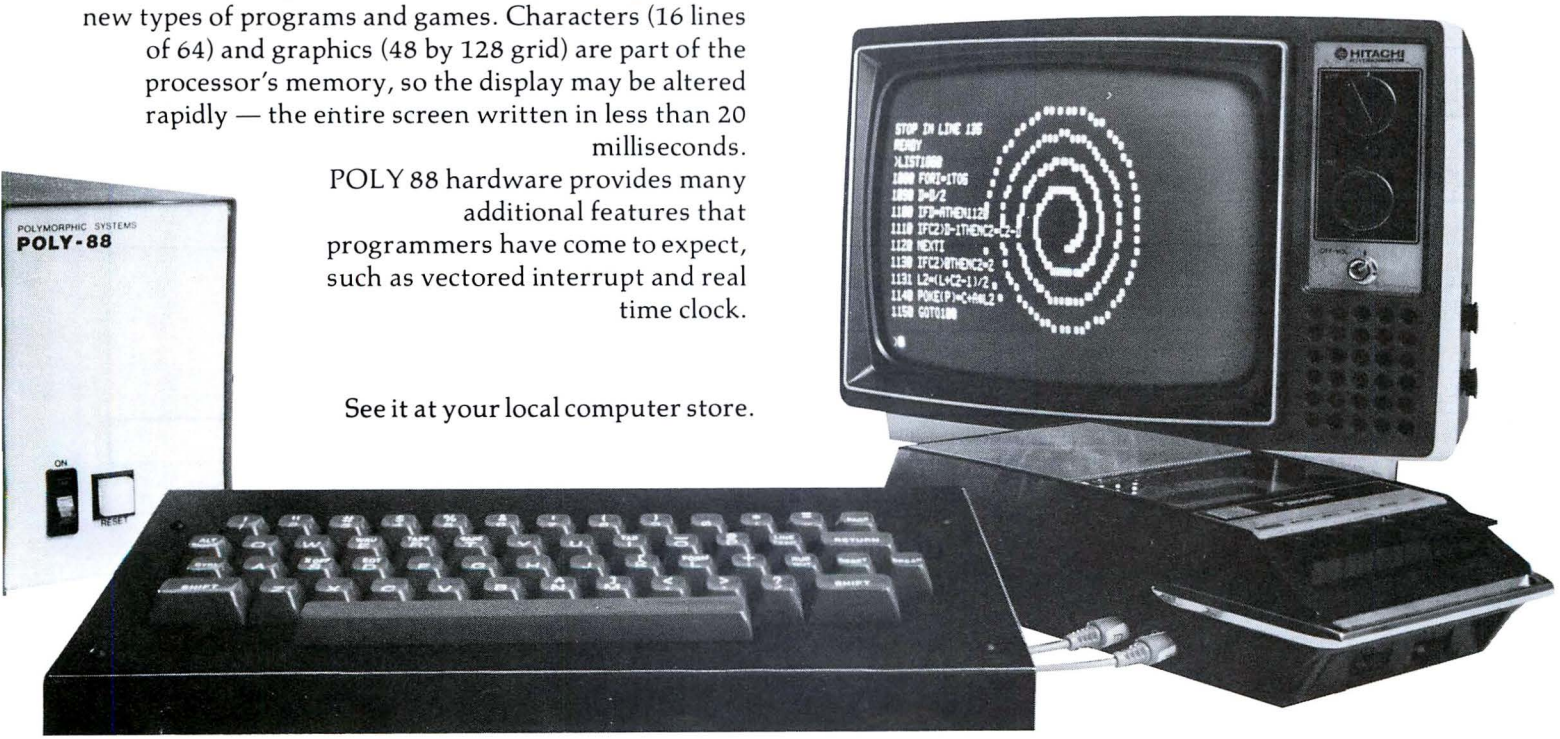
SYSTEM 4 — The complete kit. It includes system 3 *and* TV monitor, keyboard and cassette recorder with all necessary cables and connectors. $1350.

SYSTEM 7 — System 4 assembled, tested and ready to run. $1750.

ACCESSORIES — 8K RAM kit, $300. Assembled $385
POLY I/O Ideaboard, hardware prototyping kit board. $55.
Analog Interface (1 channel) kit. $145.

Prices effective until January 15, 1976. Prepaid orders shipped postpaid.

PolyMorphic Systems
737 S. Kellogg, Goleta, CA 93017
(805) 967-2351



ABSOLUTE MAXIMUM RATINGS

V_{IN} . . . . . $V_1 + 0.3$ V, $V_2 - 0.3$ V
$V_1 - V_2$ . . . . . 16 V
V_1 . . . . . 8 V
Current at V_{IN} . . . . . 1 mA
Voltage on any pin . . . . . $V_1 + 0.3$ V, $V_2 - 0.3$ V
V_{REF} . . . . . V_1

Operating Temperature . . . . . 0 to 70°C
Storage Temperature . . . . . –65 to 125°C
Power Dissipation* . . . . . 450 mW

*Device mounted with all leads welded or soldered to PC Board, Derate 6.3 mW/°C above 25°C.

ELECTRICAL CHARACTERISTICS

All DC parameters are 100% tested at 25°C. Lots are sample tested for AC parameters to assure conformance with specifications.

		CHARACTERISTIC		MIN	TYP	MAX	UNIT	TEST CONDITIONS V_1 = +5 V, V_2 = –5 V, V_{REF} = 2.000 V T_A = 25°C, C_{INT} = 0.068 µF, C_{strg} = 0.1 µF
1	GENERAL		Linearity			0.1	% rdg	
2	GENERAL		Noise		0.3		LSB	Peak-to-Peak noise apparent when going from one steady reading to another.
3	GENERAL		Zero T. C.		10		µV/°C	$0 \leq T_A \leq 70°C$
4	GENERAL		Gain T. C.		15		ppM/°C	$0 \leq T_A \leq 70°C$
5	GENERAL		NMR Normal Mode Rejection	36			dB	f_{series} = 60 Hz, f_{osc} = 24 kHz
6	INPUT	I_{IN}	V_{IN} Input Bias Current		7		pA	T_A = 25°C
7	INPUT	I_{IN}	V_{IN} Input Bias Current		90		pA	T_A = 70°C
8	INPUT	I_{REF}	V_{REF} Input Bias Current		7		pA	T_A = 25°C
9	INPUT	I_{REF}	V_{REF} Input Bias Current		90		pA	T_A = 70°C
10	INPUT	f_{CLK}	Clock Frequency		30		kHz	
11	INPUT	D.C.$_{CLK}$	Clock Duty Cycle	30/70		70/30	%	
12	INPUT	I_{INL}	Clock Input Current Low			–1	mA	V_{INL} = 1.0 V
13	INPUT	I_{INH}	Clock Input Current High			1	mA	V_{INH} = 4.0 V
14	OUT	V_{OL}	All Outputs			0.4	V	I_{OL} = 1.6 mA
15	OUT	V_{OH}	Sign, Digits	2.4			V	I_{OH} = –400 µA
16	OUT	V_{OH}	Data Bits	2.4			V	I_{OH} = –100 µA
17	SUPPLY	I_1	Supply Current			6	mA	
18	SUPPLY	I_2	Supply Current			–4	mA	
19	SUPPLY	$PSRR_1$	V_1 Supply Rejection		0.6		mV/V	
20	SUPPLY	$PSRR_2$	V_2 Supply Rejection		1.4		mV/V	

Typical values are for Design Aid Only, not guaranteed and not subject to production testing. **ICBG**

LD 130 SPECIFICATIONS

A STAR IS BORN

TDL's STARTLING NEW SYSTEM MONITOR BOARD

Replaces 4 basic boards in your system.
Integrates their functions fully into a single module.
Radically simplifies the development of your system.
Adds more versatility and performance than you'd believe possible.

- **Fully S100 bus compatible** • Power on start-up so no front panel is needed • Wait state circuitry • Fully solder masked and silk-screened Documentation includes complete software listings of the Zapple Monitor • Minimizes interfacing problems • Like having your own resident technical expert to help set up your system • Saves space for more cards • To duplicate the SMB, you'd have to spend $500 or more on separate boards.
- **2 Serial I/O and 1 Parallel I/O port.** Both serial I/O ports may be individually configured for any baud rate between 110 and 9600, in either RS232 or Current Loop configuration. The parallel port is 8 bits in or out with a status bit. All ports directly controllable by the monitor.
- **1200 BAUD audio cassette interface.** Has its own dedicated serial port and utilizes an asynchronous phase-encoding technique for high reliability and broad speed tolerance. Most important, ALL TDL software will be available on cassette in this format.
- **2K of high speed low power RAM.** Uses 1Kx4, 350ns static RAM chips, and does double duty for either add-on monitor routines (for your own special I/O) or program workspace.
- **2K Zapple Monitor in Masked ROM** gives you perfect system control direct from the console. It has 27 Commands (including 3 user defined) and offers program debug routines.

Let TDL's System Monitor Board star in your computer system. Hurry, use coupon below to order, or call (609) 921-0321

TECHNICAL DESIGN LABS Z80 — SYSTEM MONITOR BOARD — Kit $295 — Assembled & Tested $395

TECHNICAL DESIGN LABS RESEARCH PARK Building H 1101 State Road Princeton N.J. 08540

Please send your SYSTEM MONITOR BOARD. ☐ ASSEMBLED & TESTED @ $395. (N.J. residents add 5% sales tax)
☐ KIT ONLY @ $295.

Name _____ Street _____ Zip _____
City _____ State _____
I enclose ☐ Check ☐ Money Order ◇ Charge Card Data:
☐ BankAmericard Exp. Date _____
☐ Master Charge Exp. Date _____
Signature _____
☐ Send COD. I enclose 25% deposit
☐ Send your FREE CATALOG.

TECHNICAL DESIGN LABS
Research Park Bldg. H
1101 State Road
Princeton, N.J.
08540

CIRCLE INQUIRY NO. 32

When the U/D is "Down" the U/D switch is connected to V_{REF} and the current to this Integrator is:

$$I_1 - I_3 = \frac{V_{REF}}{R_1} - \frac{V_{REF/2}}{R_1} = \frac{V_{REF}}{2R_1}$$

If there is an error voltage developed in any of the amplifiers the V_{AZ} voltage will reflect this and change such as to eliminate any error current into the integrator null. Assume that we have a zero error current of 1 μa flowing into the null point. V_{AZ} will become V_{REF} per -1mv in order to null the integrator during Auto-Zero. In this "Up" condition the current to the integrator could become:

$$I_1 + I_3 + I_{ERROR} = \frac{0}{R_1} - \frac{V_{REF/2} - 1mV}{R_1} - 1_{\mu a} = \frac{-V_{REF}}{2R_1} + 1_{\mu a} - 1_{\mu a} = \frac{-V_{REF}}{2R_1}$$

During the "Down" interval, 1 μa more current flows into the R_3 mode, and the current to the integrator is

$$\frac{V_{REF}}{2R_1}.$$

You can see how nicely this system nulls out any amplifier or resistor errors so long as no parameter changes occur during the sampling interval. The Auto-Zero interval is of sufficient duration to insure that V_{AZ} will be well established.

Prior to the start of the Measure interval, the integrator output (which had been cycling around -1V) is brought back to analog ground, the comparator threshold. The system is now ready for a conversion.

The "Quantized Feedback" conversion system is characterized by a single phase Digitization interval in which a digital control system feeds back quantized units of charged in response to the sampled state of an analog comparator. These quanta of the charge balance being supplied to the integrator by the analog voltage. The magnitude

$$\left(\frac{V_{REF}}{2R_1} \cdot \frac{6}{F_{CLOCK}} \right)$$

of the Quantized charge being fed back and its sign (+ or $-$) arise from the fact that the control logic has two U/D duty cycles available during the Measure interval as shown in Figure 3.

The U/D logic is 'Up' one clock cycle and 'Down' 7 cycles for a high comparator output in the clock cycle preceding a set of eight cycles. This will be designated as duty cycle "A." With a low comparator output in clock cycle number 7 the U/D logic will be 'up' for 7 cycles and 'down' for one cycle in the following eight clock cycles. This is duty cycle "B". The effect of these two reference current duty cycles on the integrator output is shown in Figure 3. It can be seen that the 'up' state of the U/D logic drives the integrator output voltage up. The up/down BCD counter increments by each clock pulse when the U/D logic is 'up' and decrements by each clock pulse when the U/D logic is 'down'. Consequently, the net count goes up 6 counts for a 'B' duty cycle and down 6 for an 'A' duty cycle.

Input polarity is determined by the first appearance of two consecutive duty cycles of the same type. The control logic would determine the analog input to be negative if two 'A' duty cycles occur in succession and positive if two 'B' duty cycles occur in succession.

Since the counting process is done by increments (or decrements) of 6 during the measure interval, a short override interval is required at the end of the Measurement to "fine tune" the count to the nearest LSB. This occurs within the first 32 clock periods of the AZ interval. The determining equation of the LD130 is given as Count =

$$2000 \frac{V_{IN}}{V_{REF}}.$$

While this seems to imply that the output is absolutely defined by this equation; however, there is a possible 5% variation in the LD130 chip resistor values and therefore V_{REF} must be adjustable over a 5% range. This equation does indicate that with a $V_{REF} = 2.0$V the output will read directly in millivolts of analog input with 1.000 Volt full scale.

Following the count correcting override sequence; the contents of the BCD counters and sign flip-flop are loaded into the internal latches. Counter states of less than 80 or greater than 999 are decoded as underrange or overrange conditions respectively. The presence of an out-of-range signal gates a single pulse (one clock period) to the SIGN/UR/OR output during either D_1 or D_2 digit time (D_2 identifies overrange, D_1 underrange). The overrange condition also provides a visual signal by holding the digit strobe outputs low during the Measure interval. This holds the display off for ⅔ of the sampling interval giving a blinking effect. The BCD data stored in the latches is continuously scanned every 24 clock periods (8 clock times per digit). Sign information appears at the SIGN/UR/OR pin coincident with the D_3 strobe. Interdigit blanking of the Digit Strobes is achieved by taking one full clock period from both the leading and trailing edges of the strobes. Thus the digit is on 6 clock periods while the BCD data for that digit appears for the full eight clock periods. Figure 4 shows the Data Output Timing.

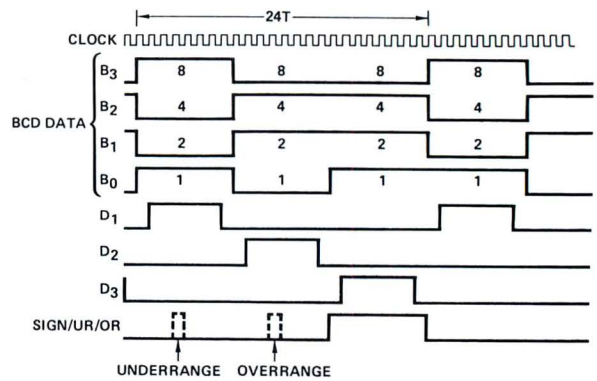
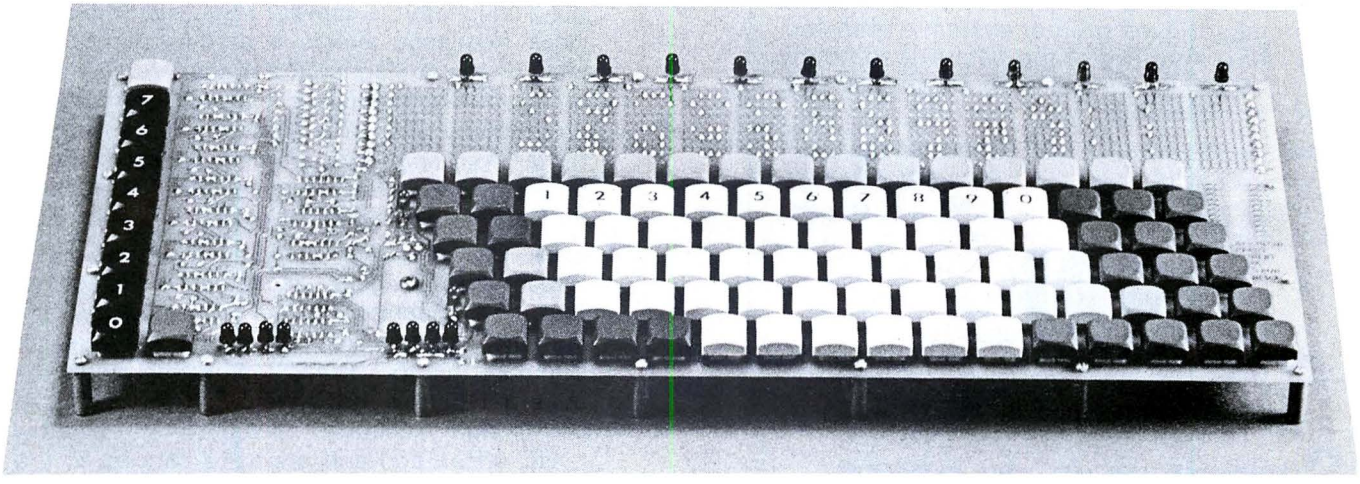


Figure 4. Data Output Format (Output = 769)

Versatility
CEI/Custom Keyboards have it



What goes on these keys is up to you

Our aim at CEI is to provide you the hobbiest, with a keyboard system that can perform any task you could possibly demand. Diode encoding coupled with one switch register encoded key allows complete latitude in key definition. Both positive and negative going strobes are provided for interfacing. Visual indication of keyboard status is provided by 22 LED's, 13 of which can have user defined meaning. Shift and shift lock are provided

Unliminted versatility w/100 Keys

90 are encoded off of a diode matrix and are fully user defined.
1 - is encoded by an 8 switch register to allow easy redefination
1 - on board reset key

Diode encoding allows a multitude of encoding techniques. ECIBIC, ASCII, BAUDOT, RADLX 50, FIELDATA or codes of your own design. Complete documentation

Computer Electronics Inc.
37433 Centralmont Place
Fremont, Calif. 94536
Phone (415) 796-5760

☐ Send ____ KB3A Kit. Supplied with all switches, diodes, PC board, TTL-IC's, +5v. regulator, connecting ribbon cable $95.95

☐ Send ____ KB3B Kit. Same as above except without diodes, TTL-IC's, +5v. regulator & ribbon cable. $49.95

☐ Case for above $29.95

NAME _____
ADDRESS _____
CITY/STATE/ZIP _____
NOTE: Calif. residents add 6% sales tax

CIRCLE INQUIRY NO. 5

Now that we have seen how the LD130 operates, let's discuss the advantages of Siliconix's "Quantized Feedback" conversion system particularly in comparison with the most commonly used integrating A/D system — Dual-Slope. These include:

1. Bipolar Conversion: Auto-Polarity is a natural outcome of a system which uses + and − reference currents. This is accomplished with a single reference voltage and a single full-scale adjustment.
2. One-Bit Wide Zero: Zero occurs naturally as a number between minus full-scale and plus full-scale in 'Quantized Feedback'. It then has none of the Dual-Slope zeroing problems which require special zeroing currents and counter delays leading to a "dead-band" around Zero of several bits.
3. Less Frequency Drift Sensitivity: "Quantized Feedback" does the conversion process while the analog input is applied. Thus, any drift in the clock frequency affects both the charge supplied by the analog input and the reference currents equally minimizing frequency drift as an error source. The Dual-Slope system integrates the analog input and reference currents during different time intervals making short term clock drift a possible source of error.
4. Less Sensitive To Comparator Errors: The "Quantized Feedback" technique only requires that the comparator resolve one part in six during the count-correcting override sequence. Compare this to a Dual-Slope which would require a 1 part in 1000 resolution for comparable 3-digit accuracy. Obviously comparator noise, offset, and drift are not a concern to the Siliconix system.
5. Fixed Conversion Time: The measure interval is fixed by the Time-Base counter. A data aquisition system once synchronized to the LD130 will not be tied up waiting for the new data. The Dual-Slope system has a conversion time proportional to the random analog input. This is particularly important to microprocessor users.

Next month we will conclude this article with a discussion of Operating Conditions, System Applications, and Microprocessor Interconnections.

INTERFACE AGE NEEDS YOUR VOTE

Help INTERFACE AGE determine the type of articles you want to see published in the future by casting your vote for the article or articles you liked best.

Feedback will provide encouragement to authors and will help make the INTERFACE AGE the microcomputer magazine of the industry.

Microcomputer Design Aides

By Terry Benson
Applications Engineer
Intel Corporation

Learning to use and program microcomputers is becoming as important today as learning logical design fundamentals was a few years ago. As time goes on, use of microcomputers will eventually increase to a point that may even approach that of the pocket calculator. Learning about microcomputers will be as critical then as learning the multiplication tables is today.

In order to simplify the educational process, with the proliferation of microprocessors, more and more companies are producing "learning tools" and educational material supporting these microprocessors. Companies are also incorporating microprocessors into "hobby" computers. This makes it possible for many of us to have a personal computer — but how do we use it effectively?

Many of these educational tools are dedicated for just that purpose and, consequently, are not practical for use as a computer system. On the other hand, many of the hobby computers, which may be functional as a microcomputer system, assume a working knowledge of computers and thereby make it difficult for the beginner to effectively use the microcomputer. With these thoughts in mind we can consider some desirable features that should be included in a low cost design aid:

1) Newcomers to microcomputing should be able to learn how to use the microcomputer effectively.
2) For long term benefits, the design aid should be able to be used as a fully functional microcomputer system.
3) Most importantly, it should be convenient to debug programs and verify that the microcomputer system performs as desired.

USEFUL FEATURES OF A DESIGN AID

The novice may have had little, if any, formal computer programming training and may be struggling to learn the programming techniques that will make his microcomputer useful to him. In many cases he will want to single step through a program to see what happens within the microprocessor when the program decrements a register, gets data from an input port, or pushes data onto the stack. While features such as this are helpful in learning how a program functions, they can also be useful in debugging a program that a more experienced programmer may write. If the program doesn't work (and I imagine you've seen many of those), it is often helpful to be able to single-step through the suspected portion of the program while checking the values within the microprocessor registers. Whether you are learning about microcomputers or debugging a program, the ability to load and run the program in RAM (read/write memory) is essential. This allows you to make immediate program changes without having to erase and reprogram a PROM.

If these capabilities are available, the microcomputer system will be more useful to the user for a longer period of time. In order to highlight the importance of some of these features, let's discuss a typical program development cycle.

DEVELOPMENT CYCLE

The flow chart in figure 1 illustrates the steps that are involved in getting a microcomputer program working. These steps will be performed whether you are writing a game, designing an automated test system, implementing a controller, or designing any other microcomputer application. In the development flow chart as shown, it is assumed that the designer has a working microcomputer system. (Although this is not always a safe assumption, the hardware debug phase will be left for future discussions.)

Writing the program, block 1, may also consist of assembling or compiling the program into a machine readable format. In the lower cost design aids, the translation is performed by hand. After conversion, the machine language must be entered into the microcomputer system, block 2. The machine language can be entered in several different ways: program a PROM, read a paper tape into RAM, load RAM from toggle switches or hexedecimal switches or from a keyboard, etc.

In block 3, we begin to check out — or debug — our program. One optimistic approach is to start the program at the beginning and hope that it runs to the end. If it doesn't perform as intended, then you probably will follow the pessimist's approach — single step all or portions of your program. In fact, blocks 3 and 4 will probably require the majority of time in the overall software development cycle.

As each one of the bugs is found, a correction must be made to the program (block 5) and the verification cycle (blocks 3 and 4) repeated. The changes can each be entered in any one of the ways that the original program was entered, but it is during this correction phase that it is most desirable to have the program residing in RAM so changes can be easily implemented. Having

the ability to specify the affected memory address and immediately entering the new or corrected data will simplify the entire debug cycle. If the program is operating in PROM, the PROM must be erased (or discarded) and reprogrammed for one more try — a rather time consuming operation if it must be done for each change.

After blocks 3, 4 and 5 have been reiterated enough times to have a properly running program, the program will have to be saved (block 6). Several alternatives for saving the program are available: the power can be left on (this might be a bit precarious); the object code can be stored on paper tape or a magnetic media; or, in cases where the program will be operating in a dedicated system application, a PROM can be programmed.

Through all of this we have been using the microcomputer system as a development system. During this development cycle, several desirable features have been mentioned. The ability to single step your program and observe the affect of each instruction can be invaluable in finding and correcting a program bug. The fact that RAM locations can be easily changed also enhances the program development cycle. One very desirable feature is the ability to copy the data in RAM ***directly to*** a PROM — a ***built-in*** PROM programmer.

A LOW COST DESIGN AID

Few microcomputer systems have all of these desirable features incorporated into a single product. One product that does have these features is the Intellec* PROMPT 80 System*, figure 2, recently introduced by Intel Corporation. The PROMPT 80 Personal Programming Tool is a self-contained microcomputer development system which supports the design of any 8080 system. It is a fully-assembled microcomputer that can be used for learning, debugging, and running 8080 programs.

The PROMPT 80 System uses an integral calculator-like hexedecimal keyboard for entry of machine language programs. (A teletype writer or CRT terminal may easily be added by simply connecting the appropriate wires.) The system is based on the SBC 80/10 Single Board Computer and includes 1K bytes of RAM, 3K bytes of ROM and 24 programmable parallel I/O (TTL) lines. A PROM programmer for the 8708/2708/2704 erasable PROMs is also included.

The following is a summary of the features that will facilitate both learning and debugging 8080 programs:

1) ***Register Display*** — all 8080 registers can be displayed, even while single stepping.
2) ***Modify Register or Memory*** — any 8080 registers or any of the 1K RAM locations can be modified.
3) ***Single Step*** — allows register data to be displayed (and modified for debugging purposes.)

*Intellec and PROMPT are trademarks of Intel Corporation

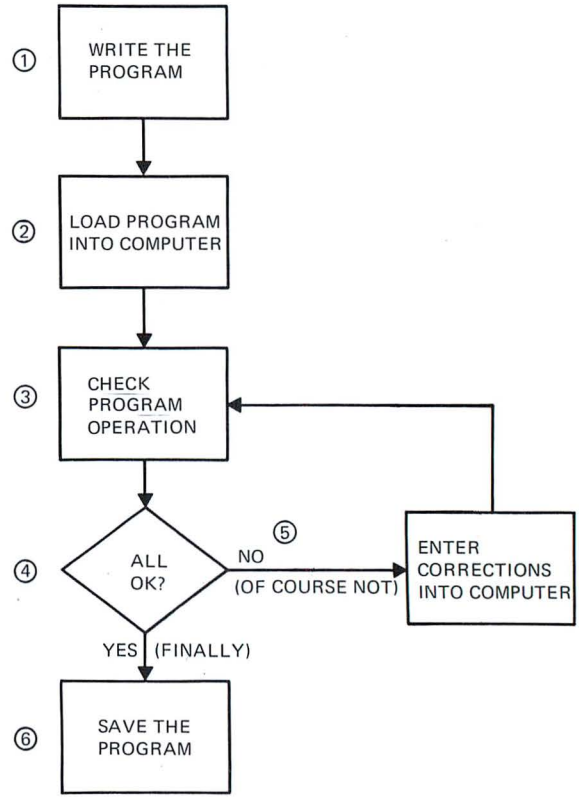


Figure 1. Program Development Flow Chart

4) ***Break-points*** — a program operating in RAM can be started and up to two break-points (program locations where you desire the program to stop) can be set.

5) ***Program EPROM (2708)*** — PROM programmer allows data to be copied from RAM to EPROM. (Data can also be transferred from EPROM to RAM.)

6) ***Hex Calculator*** — useful for adding and subtracting in the unfamiliar base 16.

7) ***Built-in I/O Ports*** — an 8-bit input port and an 8-bit output port are implemented on the control panel. These two ports along with 8 additional I/O lines are available on an I/O connector for external interface.

8) ***Serial Port*** — teletype of CRT terminal can easily be added.

9) ***Expansion*** — using a modular card cage (Intel SBC 604) and other SBC 80 expansion boards, memory and I/O can be expanded as required.

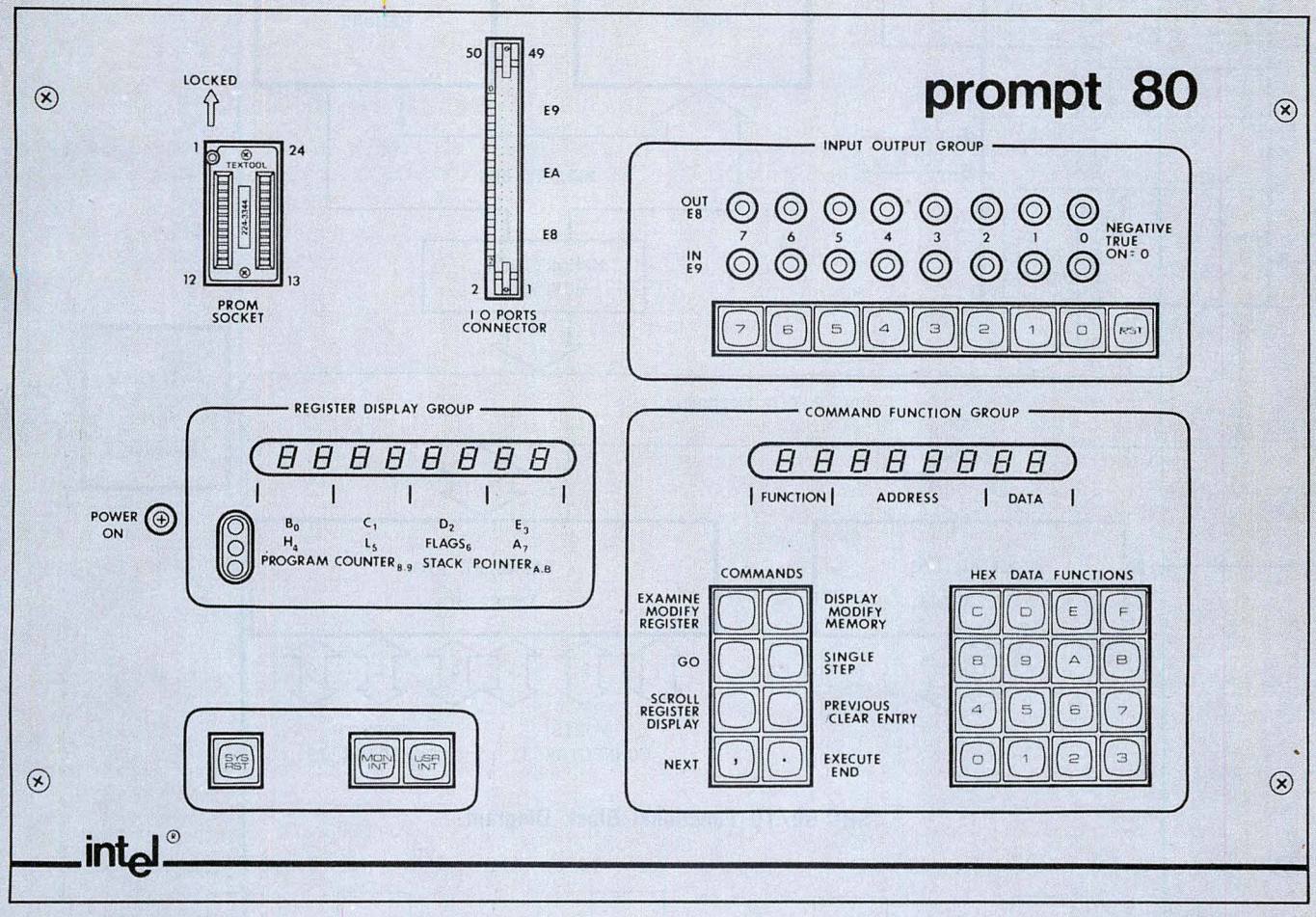


Figure 2. Panel Operation

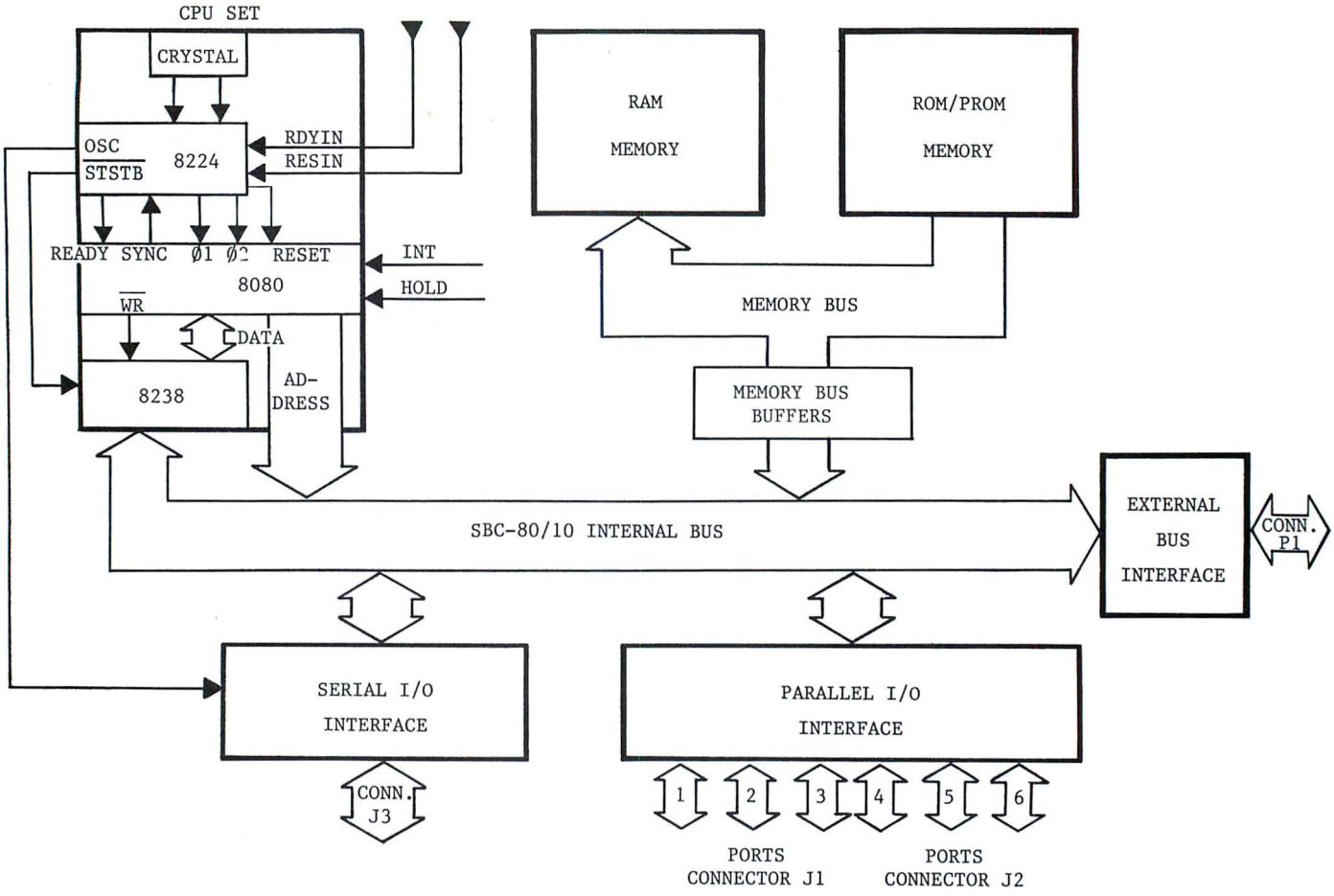
ADDITIONAL SUPPORT

A major portion of those users who will invest in a low cost microcomputer system will have had little or no experience in microcomputing. For this reason, additional design and tutorial information is included with the PROMPT 80 System. Part of the tutorial material consists of programming examples which are stored in one of the extra EPROM's that can run on the PROMPT 80 System while illustrating its uses. In conjunction with the design library is a comprehensive tutorial manual which explains microcomputing and programming concepts with novel MICROMAP* diagrams that explain the relationships among instructions for memory, input/output and internal CPU operations. A full set of reference schematics are also included with the PROMPT 80 System.

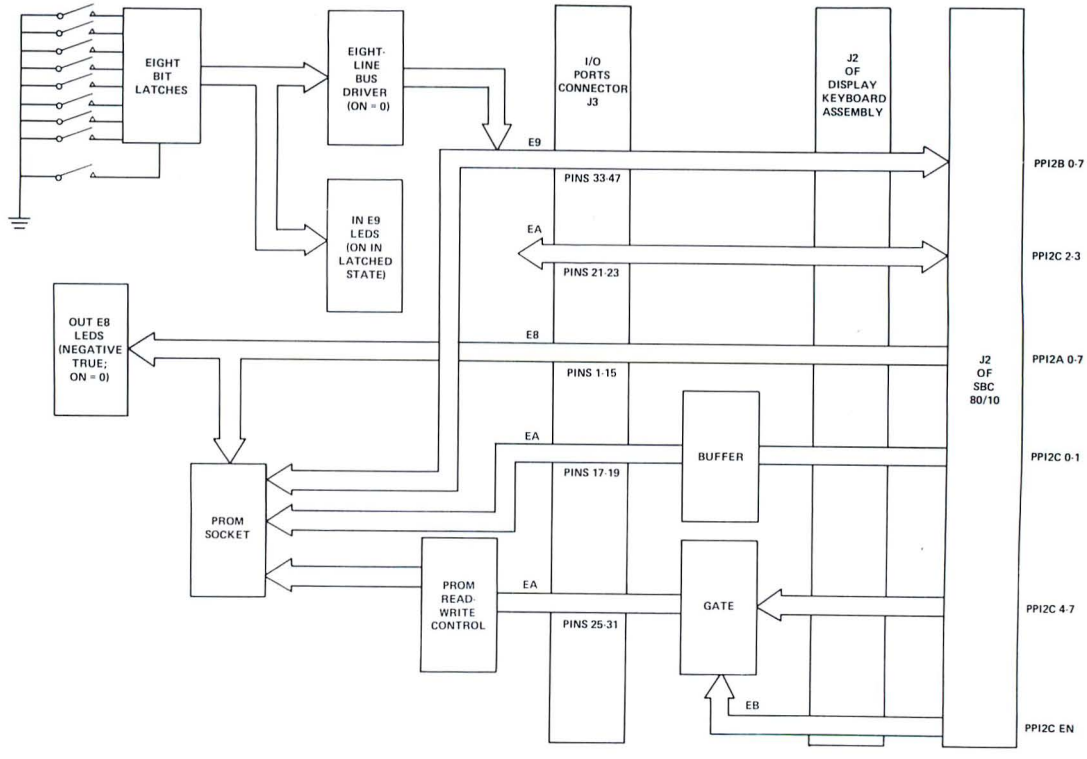
The Intellec PROMPT 80 Personal Programming Tool is now available at $1495. The price includes complete assembly and testing, two 8708 EPROM's, three 8308 ROM's containing the system monitor programs, along with the design library. Optional accessories include a cable for connection to the Intellec Microcomputer Development System as a specialized PROM programmer; a cable for connection to the serial port for addition of a terminal; expansion modules including memory boards, I/O boards, combination memory and I/O boards, card cage and backplane modules, a diskette controller module, and diskette systems.

In summary the PROMPT 80 System offers the beginner a means of easily learning microcomputing and provides a convenient method for debugging programs. The fact that it can be expanded both in capacity and peripheral support makes it a cost effective base for building a complete microcomputer system.

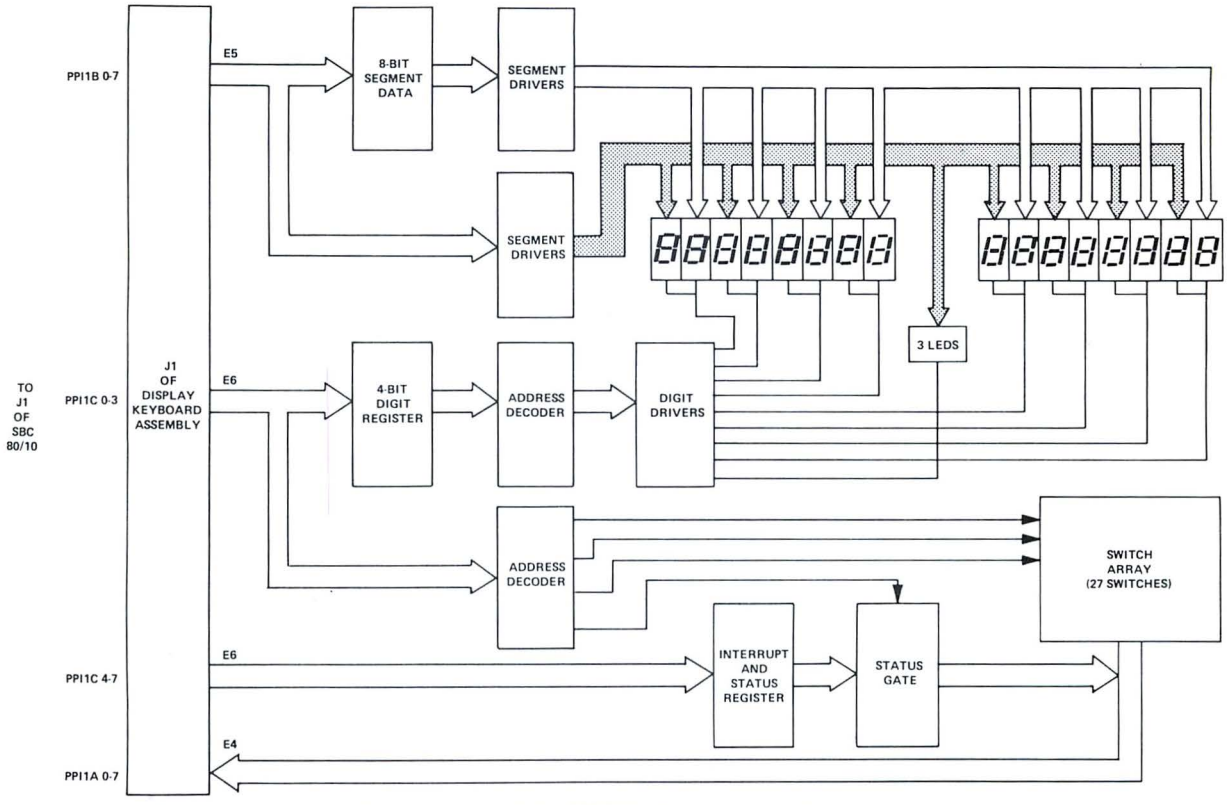
*Micromap is a trademark of Intel Corporation



SBC-80/10 Functional Block Diagram



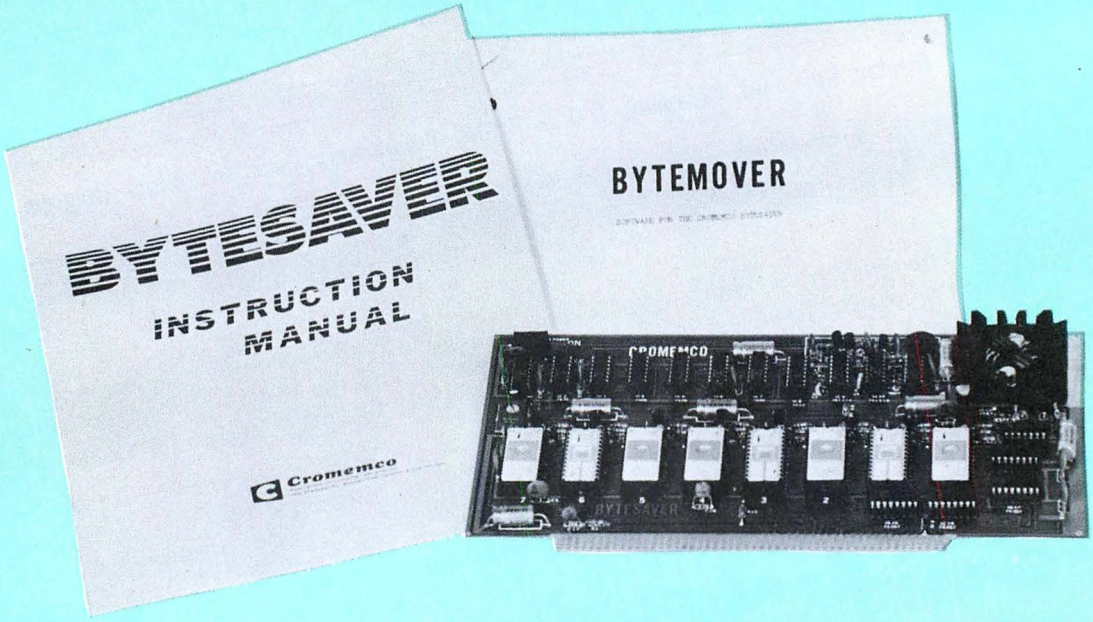
Display Keyboard Assembly, Display & Keyboard Functions, Block Diagram



Display Keyboard Assembly, I/O & PROM Programming Functions, Block Diagram

CARD-OF-THE-MONTH

CROMEMCO BYTESAVER™



by Roger Edelson

JANUARY

Are you tired of reentering your favorite program everytime you turn off power? Does the thought of waiting 15 minutes for your BASIC to read in fill you with dread? Would you like to have your assembler/editor/monitor available within one second after turn-on? If you answered yes to any of these questions, then you ought to consider this month's card — the CROMEMCO BYTESAVER™.

If I could purchase one board (after some memory and an I/O card) the BYTESAVER would be my choice. The board is both an economical PROM (Programmable Read Only Memory), with the capacity for a full 8K bytes of storage, and a low-cost means of storing your programs in PROM. The BYTESAVER™ is an S-100 bus compatible memory board. Space is provided for eight 2708 U.V. erasable PROMs, for a full 8K bytes of memory. Using the 2708 allows your 8080 machine to operate no wait states, however, provision for a wait state is provided should you desire to purchase slower 2704 or 2708 PROMS.

The BYTESAVER™ also provides you with your own low cost PROM programmer. Using the software provided programs may be transferred from non-permanent RAM memory to the permanent PROM memory in the BYTESAVER™. Once your program is stored in PROM it's protected from power turn-offs, either accidental or intentional. The PROMs used are U.V. erasable and may be used again and again. Your program may be run either directly out of PROM, or using the software provided, it may be transferred to RAM memory beginning at any 1K segment selected. The software provided with the BYTESAVER™ controls both the transfer of data from the PROMs or to the PROMs. It is designed so there is no need for a keyboard. Just set the computer sense switches as instructed in the documentation. Transfer of RAM content to PROM takes less than a minute, and transfer of the 8K from PROM to RAM is complete in less than one second. The software controls the computer lights to provide verification of complete and accurate transfer of memory content.

As far as the design goes, the BYTESAVER™ is not particularly complex. The board is S-100 bus (ALTAIR/-IMSAI) compatible, and contains space for eight 2708/2704 PROMs. In order to program the eight 2708 PROMs a high voltage (+30V) supply is required, and this is generated from the regulated +5V supply by a DC to DC Converter. Switching (Protect/-Unprotect) is provided to disable this supply to avoid inadvertent overwrite of the memory in the case of a user program malfunction. The DC to DC converter appears to be a standard blocking oscillator configuration with a very simple feedback regulator. I say appears because the schematic does not show the internal arrangement of the pulse transformer used in the converter. Since there is hardly anything magic about a blocking oscillator it seems strange that Cromemco doesn't help those of us who might be able to service our own boards by including the transformer arrangement.

Distribution of the program voltage to the 2708s is by means of discrete transistors controlled by 7406 open-collector inverting buffers. A schematic of the BYTESAVER™ is shown in Figure 3. Control of the $\overline{CS}$/WE pin (which requires +12 V during WRITE) is done by a 7406 and a discrete transistor which allows

this pin to float up to +12V. During READ operation the $\overline{CS}$/WE pin is either a ground or +5V, as the +12V supply is disabled.

While we are discussing the schematic let me suggest to Cromemco that the 7406 is an inverting buffer and I would prefer it to be shown that way, always. It is difficult to follow a logic diagram if the inverting sign is left out, even if it is desired to show logical flow. If necessary an inverting circle could have been placed at the $\overline{CS}$/WE input pin at the 2708. A further help would have been to have the component designation on the schematic also. These are all minor quibbles, however, and they do not detract from the board's usefulness and operation.

Returning to the BYTESAVER™ circuit, the data bus lines are all buffered by 74367's (the standard 54/74 equivalent to the ubiquitous 8T97 series). Address decoding for the highest 8K of memory (A13-A15) is provided by jumper connections on the board. In order to provide addressing in this fashion using a three input NAND gate (rather than the slightly more expensive and easier to use octal decoder and dip switch) inverted address signals must be made available. Three inverters of a 74LS04 are used to derive $\overline{A13}$, $\overline{A14}$, and $\overline{A15}$. Since there were three inverters left over, these were used to buffer the A10 through A12 signals to the 7442. For this reason the low order address selects the output 7 pin. A bit strange but it works, and very well.

A 74123 dual retriggerable one-shot is used to provide the proper pulse width (N/MS) needed for programming the PROM. This pulse is also used to produce a wait state during PROM programming. As mentioned earlier, it is possible to jumper connect the capability for one wait state if slower PROMs are used. This is done at the input to the three input NAND gate whose output drives the clear pin on the second half of the 7474 'D' type Flip-Flop.

The board is high quality G-10 material with tinned solder pads and gold plated edge board connectors for reliability. The kit is fully socketed and assembly is suggested to be about one evening; it doesn't even take that long. A high quality soldermask contributes to the ease of assembly and minimizes the chance of solder splashes. The assembly instructions are explicit enough and the board is adequately screened to provide easy identification of component placement. Cromemco recommends no diode be installed in the position below QO. This will prevent inadvertant rewrite of the BYTEMOVER™ stored in PROM O. The assembly instructions also warn that a pair of IC regulators look physically similar, BUT are NOT interchangeable. Also a nylon screw is provided to prevent shorting a regulator to ground.

One of the niceties of the BYTESAVER™ is the capabilities provided by its software. When you purchase a BYTESAVER™ with one 2704 PROM, the Bytemover software is pre-programmed in that PROM. For your information and assistance, Cromemco has allowed INTERFACE AGE to publish the BYTEMOVER™ software. This program is included at the end of this article.

The PROM containing the Bytemover software is normally inserted into PROM location Zero on the Bytesaver board.

The Bytemover software can be used to program a PROM in any of the PROM locations on the Bytesaver board. The Bytemover software can also be used to transfer programs from PROM to RAM.

The operation of the Bytemover software is controlled by setting front panel sense switches on any S-100 bus compatible computer. However, to use the Bytemover software there must be at least one RAM board in the computer beginning at location Zero in the memory map. Furthermore, this RAM board must be unprotected for proper execution of the Bytemover software.

Software can be loaded into a 2704 or a 2708 in as small increments as you desire provided it is added to previously unused areas in that PROM.

This is done by first using Bytemover to move the current contents of the PROM down to RAM which corresponds to the unused portion of the PROM and finally using Bytemover again to reprogram the PROM with the new software.

Although the entire PROM must always be programmed, it never hurts to re-write the same information over again. And, of course, an erased PROM in which all bits are "1" may be programmed at any time.

In general, it is OK to write a "1" over a "1", a "0" over a "0", or a "0" over a "1", but to write a "1" over a "0" the PROM must be completely erased.

If the PROM to which you want to add software is PROM zero on the Bytesaver board, turn off AC power to the computer and install a 1N914 diode just below QO. Turn the power back on and move Bytemover to RAM zero. Add the new software to an area of RAM which corresponds to an unused portion of PROM zero.

Re-program PROM zero by following the example in the manual supplied. Note you need not erase the PROM to do this. Turn the computer power off and remove the 1N914 diode below QO.

The Bytesaver software supplied with the board is designed to program the PROM in approximately 30 seconds. This is generally a sufficiently long period of programming time. However, to be completely within the manufacturer's specifications, the PROM should be programmed for two to three minutes.

If you wish to program your PROMs for longer than 30 seconds, the Bytemover software may be easily modified. Simply change the contents of memory location 77H (Hex) from 40H to 00H. Now you must manually time the programming operation and depress the stop switch at the end of the operation.

Before using the Bytesaver, you must install three jumper wires to set the location of the Bytesaver in memory. This adjustment is shown in Figure 1. The assembled Bytesaver comes with A_{13}, A_{14}, and A_{15} each tied to the corresponding 'HI' pad to position the board at the very top of memory. In the following instructions it is assumed this is the jumper connection used.

1. Turn off all power to the computer and plug in the Bytesaver board.
2. Be sure the program power on the Bytesaver is turned off (program power switch in the down position).
3. Turn on the computer. Raise the reset switch, the stop switch and then raise the reset switch once again to initialize the computer.

The Byte Shop reaches a new low in microcomputers. $349

The Byt-8. It doesn't have a nifty (and expensive) front panel with lots of LED's and toggle switches.

And we obviously don't have a big full-color ad.

What we do have is the lowest priced microcomputer you can buy—built around the powerful and popular 8080A microprocessor.

For $349, you get the complete microcomputer card, motherboard, power supply and chassis in kit form.

The Byt-8 S100 bus is the same one used by Altair, IMSAI and most others so you have the greatest possible flexibility in choosing memory and input/output cards.

Optional cards from the Byte Shop include 4k, 8k or 16k of Random Access Memory, 4k or 8k of Programmable Read-Only Memory, a multiple input/output card, a TV typewriter card and, yes, a front panel bootstrap card, if you want the LED's and switches.

Even the CPU is optional. We'll sell you the chassis, motherboard and power supply for $229, and you can choose your own microcomputer card—a ZPU for instance?

Byt-8. It's the new low in price, but we're aiming for a new high in flexibility, delivery and support. See the Byt-8 at your nearest Byte Shop.

BYTE SHOP™
the affordable computer store

Stores now open in: **Arizona,** Tempe; **California,** Berkeley, Campbell, Fresno, Hayward, Mountain View, Palo Alto, Pasadena, Sacramento, San Jose, San Mateo, San Rafael, Santa Barbara, Santa Clara, Santa Cruz, Tarzana, Thousand Oaks, Walnut Creek, Westminster; **Colorado,** Boulder, Englewood; **Minnesota,** Minneapolis/St. Paul; **New York,** Levittown; **Oregon,** Portland; **Pennsylvania,** Bryn Mawr; **South Carolina,** Columbia. If there's no Byte Shop near you yet, please write to Byte Inc., 1450 Koll Circle, Suite 105, San Jose, California 95112 for information on our Byt-8 system.

CIRCLE INQUIRY NO. 38

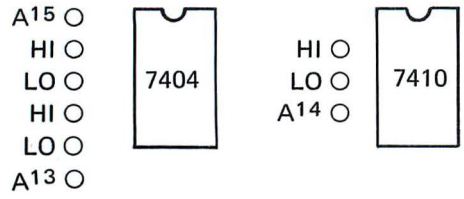


FIGURE 1. *This diagram illustrates the positioning of the Bytesaver address in memory. The preassembled Bytesaver comes with A^{15}, A^{14} and A^{13} connected to the corresponding HI terminals so memory address occurs when these lines are HI. Any or all of these address lines may be connected to the corresponding LO terminal to move the memory board lower in memory. Consequently, there are eight positions in which this board may be used.*

4. Raise the address switches A_{15}, A_{14}, and A_{13}. All other address switches should be in the down position.
5. Raise the examine switch. You are now examining the contents of the first byte of PROM in PROM location zero of the Bytesaver memory board (memory location 340 000). If the PROM supplied with your Bytesaver is in this PROM location, the data lights will read '061', the first byte of the Bytemover program.

The following examples show the steps needed to transfer and program the PROMs and the ease in which it is done.

EXAMPLE 1: Transfer the Bytemover Program from PROM to RAM beginning at RAM location Zero.

1. Raise the reset switch.
2. Depress the unprotect switch (on the Altair front panel).
3. Raise A_{15}, A_{14}, and A_{13}. Raise the examine switch. The data lights should read '061' octal.
4. Now set the sense switches for the task to be done, referring to Figure 2.
5. Push the run switch. In less than one second, the contents of PROM will be transferred to RAM. The contents of PROM are unaffected by this operation.
6. Raise the stop switch.
7. Raise the reset switch. Note that the data lights read '061.'

EXAMPLE 2: Program a 2708 PROM inserted in PROM location one. This PROM is to be programmed with the contents of the first 1K bytes of RAM beginning at location zero in memory. The Bytesaver software is still in the PROM installed in PROM location zero on the Bytesaver board.

A^{15}	Down	to transfer from Prom to Ram.
A^{14}	Down	for the transfer of 1K bytes.
A^{13}	Down	All down since we are transferring from the PROM that contains Bytemover (PROM 0).
A^{12}	Down	
A^{11}	Down	
A^{10}	Down	All down for storage to begin at location zero in RAM.
A^{9}	Down	
A^{8}	Down	

1. Raise the reset switch.
2. Depress the unprotect switch on the Altair front panel).
3. Raise A_{15}, A_{14}, and A_{13}. Raise the examine switch. The data lights should read '061' octal.
4. Raise the protect switch on the Bytesaver board (i.e. program power switch to the on position). The protect light on the front panel should turn off when this switch is raised.
5. Now set the sense switches for the task to be done.

A^{15}	Up	to program a PROM
A^{14}	Down	(always down for PROM programming).
A^{13}	Down	To select the PROM 1K higher in memory than the PROM that contains Bytemover
A^{12}	Down	
A^{11}	Up	
A^{10}	Down	All down for transfer to begin at location zero in RAM.
A^{9}	Down	
A^{8}	Down	

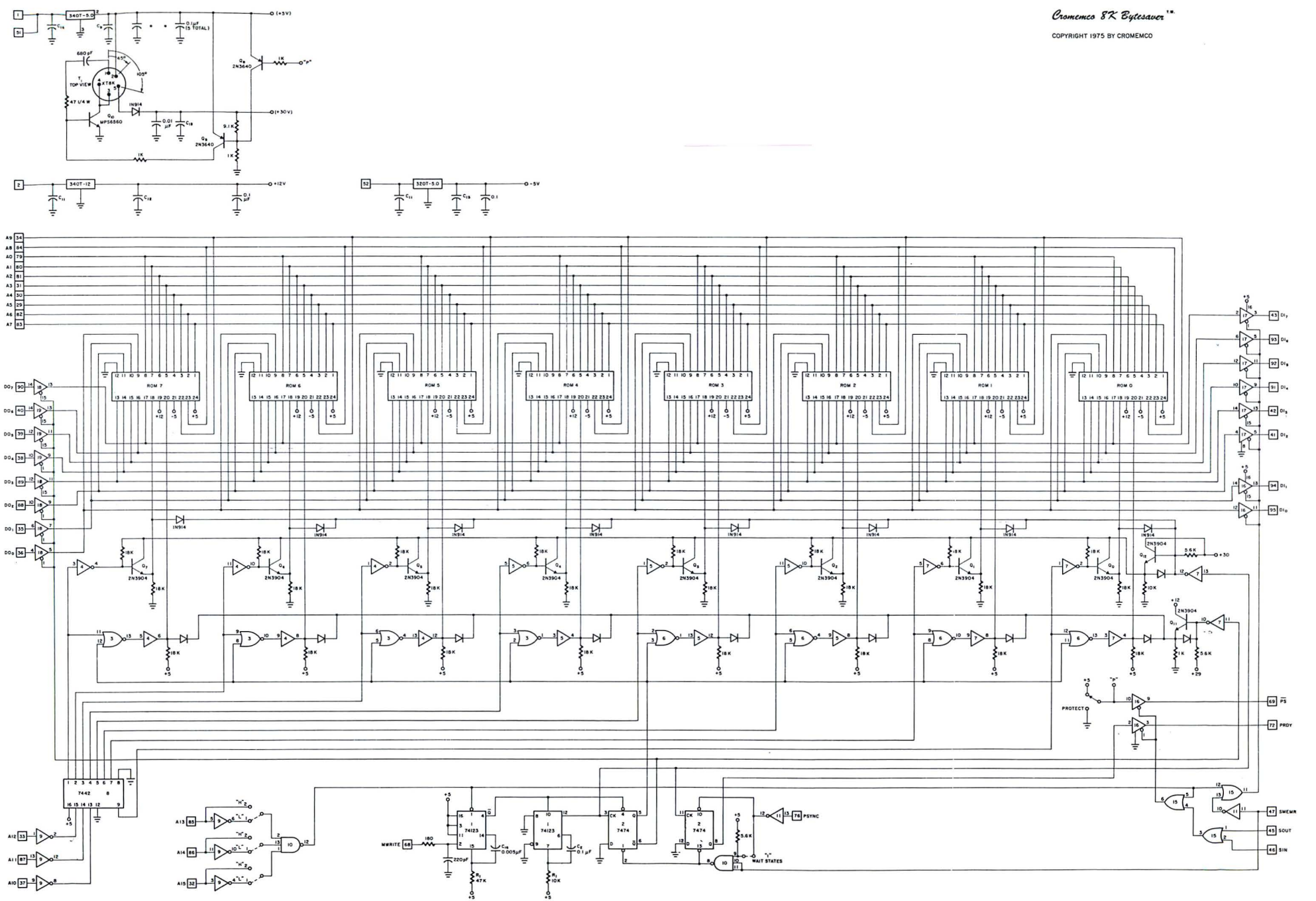
6. Push the run switch. Note that panel light A_9 is blinking at a rate of about twice per second. When this light stops blinking, the PROM programming is complete.

A^{15}	A^{14}	A^{13} A^{12} A^{11}	A^{10} A^{9} A^{8}
Up To program a PROM. Down To move from PROM to RAM.	Up For a 7K transfer. Down For a 1K transfer.	MSB LSB PROM address location in increments of 1K from the PROM in which BYTE	MSB LSB Selection of RAM address in 1K increments.

FIGURE 2. FUNCTION OF THE SENSE SWITCHES IN BYTEMOVER.

BYTESAVER™ SCHEMATIC

Cromemco 8K Bytesaver™



7. Raise the stop switch.
8. Now note the INTE light on the front panel. If this light is on, the Bytemover Verifier has verified that the contents of the programmed PROM are indeed identical to the contents of the selected 1K bytes of RAM. If this light is off, the PROM is not programmed correctly. This could be due to a defective PROM.

EXAMPLE 3: Altair 8K BASIC can be stored in seven 2708 PROMs. Given that these seven PROMs are in PROM locations one through seven of the Bytesaver board, 8K BASIC can easily be transferred into RAM using the following procedure.

1. Raise the reset switch.
2. Depress the unprotect switch (on the Altair front panel).
3. Raise A_{15}, A_{14}, and A_{13}. Raise the examine switch. The data lights should read '061' octal.
4. Now set the sense switches for the task to be done.

A_{15}	Down	to transfer from PROM to RAM.
A_{14}	Up	for a 7K transfer.
A_{13}	Down	To begin transfer from the PROM 1K higher in memory than the Bytemover program.
A_{12}	Down	
A_{11}	Up	
A_{10}	Down	All down for storage to begin at location zero in RAM.
A_9	Down	
A_8	Down	

5. Push the run switch. In less than one second BASIC will be loaded into RAM (it sure beats paper tape!). Now raise the stop switch.

EXAMPLE 4: If you do not have Bytemover in PROM, you can program a PROM with Bytemover that is stored in RAM. The Bytemover software must be loaded into RAM beginning at location zero. The Bytemover software can then be burned into a PROM using the following procedure.

1. Raise the reset switch.
2. Depress the unprotect switch (on the Altair front panel).
3. Insert an erased PROM into PROM location zero.
4. Examine location 000 240 in memory.
5. Raise the program power switch on the Bytesaver board.
6. Set the sense switches with A_{15}, A_{14}, and A_{13} up.
7. Push the run switch. When the A_9 light stops blinking, the programming is complete. The INTE light will be on.
8. Turn off PROM program power by depressing the switch on the Bytesaver board.

That's about all there is to it to use a Bytesaver. It's both convenient and easy. The new instruction manual is a great improvement over the previous one. I would have liked to have a little more information on erasing PROMs. The PROM should be about one inch away and exposed for about twenty minutes. This assumes an integrated dosage of 10W-SECS/CM^2 is required. If shorter times or lower output devices are used, I would recommend placing the PROMs in the Bytesaver and running a memory check. All bits should be '1' for correct operation. PROM erasers are available from various computer stores around the country.

The Bytesaver is a tremendous adjunct to your computer capability (I have two, one with an assembler/monitor program and the other with BASIC residing in memory). If you expect to program a greater number of different PROMs you would be wise to replace the PROM sockets with low or zero insertion force types. Either that or place the PROM in carriers. This is to reduce the strain on their leads. The Bytesaver may be purchased with the Bytemover software already stored in either a 2704 or a 2708. By the way, the 2704 is just a 2708 with a bad bit in the upper 512 bytes. 2708 PROMs can also be purchased from Cromemco, but their price is a little higher than you can get elsewhere, but the convenience may be worthwhile. Anyway if you choose to buy it, I would heartily recommend considering the Cromemco Bytesaver as your next purchase.

```
0000                    0000 * BYTEMOVER (T.M.) SOFTWARE FOR
0000                    0001 * CROMEMCO 8K BYTESAVER (T.M.)
0000                    0002 * VERSION 3. 1
0000                    0003 * SELF-RELOCATING SOFTWARE LOCATABLE AT ANY
0000                    0004 * 1024 BYTE (1K) BOUNDARY IN MEMORY
0000                    0009 * ROUTINE TO FIND ONESELF IN MEMORY
0000                    0010   SP EQU 6
0000                    0019 * DEFINE FIRST 4 BYTES IN MEMORY AS STACK
0000 31 00 00           0020   LXI SP, 0
0003                    0029 * SAVE FIRST FOUR BYTES IN REGISTERS
0003 C1                 0030   POP B
0004 D1                 0040   POP D
0005                    0049 * REPLACE BYTE 0 WITH A 'RETURN'
0005 2E C9              0050   MVI L, 0C9H
0007 F3                 0051   DI
0008 E5                 0060   PUSH H
0009 E5                 0070   PUSH H
000A 00                 0080   NOP
000B 00                 0081   NOP
000C 00                 0082   NOP
000D 31 04 00           0090   LXI SP, 4
0010 CD 00 00           0100   CALL 0
0013                    0101 * ROM LOCATION NOW IN BYTE 3
0013 31 02 00           0110   LXI SP, 2
0016 E1                 0120   POP H
0017                    0129 * RETURN BYTES 0-3
0017 31 04 00           0130   LXI SP, 4
001A D5                 0140   PUSH D
001B C5                 0150   PUSH B
001C                    0159 * STORE ROM LOCATION IN SP
001C F9                 0160   SPHL
001D 0E 00              0170   MVI C, 0
001F 59                 0180   MOV E, C
0020 69                 0190   MOV L, C
0021                    0199 * INPUT SENSE SW COMMANDS
0021 DB FF              0200   IN 255
0023 57                 0210   MOV D, A
0024                    0219 * STRIP RAM ADDRESS
0024 E6 07              0220   ANI 7
0026 07                 0230   RLC
0027 07                 0240   RLC
0028                    0249 * STORE RAM ADDRESS IN BC
0028 47                 0250   MOV B, A
0029 7A                 0260   MOV A, D
002A                    0269 * STRIP ROM ADDRESS
002A E6 38              0270   ANI 56
002C 0F                 0280   RRC
002D 00                 0290   NOP
002E 67                 0300   MOV H, A
002F 39                 0310   DAD SP
0030 2E 00              0320   MVI L, 0
0032 7A                 0330   MOV A, D
0033 EB                 0340   XCHG
0034                    0341 * ADDRESS OF ROM BEING PROCESSED IN DE
0034                    0349 * BRANCH TO TRANSFER OR PROGRAM ROUTINE
```

```
0034 E6 80      0350    ANI 128
0036 0F         0360    RRC
0037 0F         0370    RRC
0038 C6 2D      0380    ADI 45
003A 21 00 00   0390    LXI H, 0
003D 6F         0400    MOV L, A
003E 39         0410    DAD SP
003F E9         0420    PCHL
0040            0500 *  ROUTINE TO TRANSFER ROM TO RAM
0040 F9         0510    SPHL
0041 21 0B 00   0520    LXI H, 11
0044 39         0530    DAD SP
0045 EB         0550    XCHG
0046 F9         0560    SPHL  STACK CONTAINS ROM LOCATION
0047 EB         0570    XCHG  H&L CONTAIN LOOP ADDRESS
0048 11 00 00   0580    LXI D, 0
004B            0588 *  START OF TRANSFER LOOP
004B            0589 *  INCREMENT ROM ADDRESS
004B 3B         0590    DCX SP
004C            0599 *  MOVE DATA FROM ROM TO RAM
004C F1         0600    POP 6
004D 02         0610    STAX B
004E            0619 *  INCREMENT RAM ADDRESS
004E 03         0620    INX B
004F            0629 *  INCREMENT BYTE COUNT
004F 13         0630    INX D
0050 7A         0640    MOV A, D
0051 E6 04      0650    ANI 4
0053 07         0660    RLC
0054 07         0670    RLC
0055 00         0680    NOP
0056 85         0690    ADD L
0057 6F         0070    MOV L, A
0058 E9         0710    PCHL
0059 00         0716    NOP
005A 00         0717    NOP
005B            0719 *  JUMP TO 00B1 FROM TRANSFER ROUTINE
005B 3E 56      0720    MVI A, 56H
005D 85         0725    ADD L
005E 6F         0730    MOV L, A
005F E9         0740    PCHL
0060            1000 *  ROUTINE TO PROGRAM ROM
0060 00         1010    NOP
0061            1019 *  MOVE RAM ADDRESS INTO HL
0061 69         1020    MOV L, C
0062 7C         1030    MOV A, H
0063 60         1040    MOV H, B
0064            1049 *  MOVE RAM ADDRESS INTO SP
0064 F9         1050    SPHL
0065 67         1060    MOV H, A
0066 2E 6B      1070    MVI L, 107
0068            1079 *  INCREMENT RAM ADDRESS
0068 01 00 00   1080    LXI B, 0
006B            1089 *  INCREMENT RAM ADDRESS
006B 3B         1090    DCX SP
006C            1098 *  USE STAX AND POP 6 (PSW)
006C            1099 *  TO MOVE DATA FROM ROM TO RAM
006C F1         1100    POP 6
006D 12         1110    STAX D
006E            1119 *  INCREMENT ROM ADDRESS
006E 13         1120    INX D
006F            1129 *  INCREMENT BYTE COUNT
006F 03         1130    INX B
0070            1138 *  B STORES TWO CONSTANTS
0070            1139 *  # COMPLETE PASSES & IN ROM CNT
0070 78         1140    MOV A, B
0071            1149 *  # PASSES = 32 ?
0071 FE FC      1150    CPI 252
0073 3F         1160    CMC
0074 1F         1170    RAR
0075 1F         1180    RAR
0076            1198 *  SET 64 TO 0 FOR TWO MINUTE TIMER VERSION
0076 E6 40      1200    ANI 64
0078            1201 *  A=64 IF COMPLETED 32 PASSES
0078 2E 7D      1205    MVI L, 7DH
007A 85         1210    ADD L
007B 6F         1220    MOV L, A
007C E9         1225    PCHL
007D 2E 6B      1226    MVI L, 6BH
007F 78         1230    MOV A, B
0080 E6 04      1240    ANI 4
0082            1241 *  A=4 IF END OF 1024 BYTE PASS
0082 07         1250    RLC
0083 07         1260    RLC
0084 07         1270    RLC
0085 85         1280    ADD L
0086 6F         1290    MOV L, A
0087            1291 *  GO BACK TO 1090 UNLESS OVERFLOW
0087            1292 *  THEN GO TO 1380 FOR
0087            1293 *  ADDRESS SUBTRACTION
0087            1294 *  OR 2135 FOR QUITS
0087 E9         1300    PCHL
```

```
0088 00          1350  NOP
0089 00          1360  NOP
008A 00          1370  NOP
008B             1378 * ANOTHER PROGRAM PASS TO BE DONE
008B             1379 * ADJUST ROM AND RAM ADDRESSES
008B 7C          1380  MOV A, H
008C 21 00 FC    1390  LXI H, 64512
008F             1399 * SUBTRACT 1024 FROM ROM ADDRESS
008F 39          1400  DAD SP
0090 F9          1410  SPHL
0091 21 00 FC    1420  LXI H, 64512
0094             1429 * SUBTRACT 1024 FROM RAM  ADDRESS
0094 19          1430  DAD D
0095 EB          1440  XCHG
0096 67          1450  MOV H, A
0097 2E 6B       1460  MVI L, 107
0099 78          1470  MOV A, B
009A E6 F8       1480  ANI 248
009C             1489 * INCREMENT PASS COUNTER BY ONE
009C C6 08       1490  ADI 8
009E 47          1495  MOV B, A
009F             1499 * GO BACK TO 1090
009F E9          1500  PCHL
00A0             2000 * ROUTINE TO LOAD BYTEMOVER INTO ROM
00A0 DB FF       2010  IN 255
00A2 47          2020  MOV B, A
00A3 E6 E0       2030  ANI 224
00A5 1E 00       2040  MVI E, 0
00A7 4B          2050  MOV C, E
00A8 57          2060  MOV D, A
00A9 78          2070  MOV A, B
00AA E6 1F       2080  ANI 31
00AC 47          2090  MOV B, A
00AD 67          2100  MOV H, A
00AE 2E 60       2110  MVI L, 96
00B0 E9          2120  PCHL
00B1             2121 * CHECK FOR 7K TRANSFER OF ROM TO RAM
00B1 C6 1A       2122  ADI 1AH
00B3 6F          2123  MOV L, A
00B4 DB FF       2124  IN 255
00B6 E6 40       2125  ANI 64
00B8 0F          2126  RRC
00B9 0F          2127  RRC
00BA 85          2128  ADD L
00BB 6F          2129  MOV L, A
00BC E9          2130  PCHL
00BD             2133 * PROGRAMMER VERIFICATION ROUTINE
00BD             2134 * PART 1
00BD 7C          2135  MOV A, H
00BE 21 00 FC    2145  LXI H, 64512
00C1 39          2155  DAD SP
00C2 F9          2165  SPHL
00C3 2E CD       2175  MVI L, 0CDH
00C5 67          2185  MOV H, A
00C6 E9          2195  PCHL
00C7 00          2205  NOP
00C8 00          2210  NOP
00C9 00          2215  NOP
00CA 00          2220  NOP
00CB             2229 * ROM TO RAM TRANSFER STOP ROUTINE
00CB FB          2230  EI
00CC E9          2240  PCHL
00CD             2248 * PROGRAMMER VERIFICATION ROUTINE
00CD             2249 * PART 2
00CD 7C          2250  MOV A, H
00CE 21 00 FC    2260  LXI H, 64512
00D1 19          2270  DAD D
00D2 EB          2280  XCHG
00D3 2E F1       2290  MVI L, 0F1H
00D5 67          2300  MOV H, A
00D6 01 00 00    2310  LXI B, 0
00D9 E9          2320  PCHL
00DA 00          2625  NOP
00DB             2629 * 7K TRANSFER COMPLETION CHECK
00DB D6 90       2630  SUI 90H
00DD 6F          2640  MOV L, A
00DE 7A          2650  MOV A, D
00DF C6 04       2660  ADI 4
00E1 57          2670  MOV D, A
00E2 FE 38       2680  CPI 56
00E4 3F          2685  CMC
00E5 3E 00       2690  MVI A, 0
00E7 1F          2700  RAR
00E8 85          2710  ADD L
00E9 6F          2720  MOV L, A
00EA E9          2730  PCHL
00EB             2879 * ROM PROGRAMMER STOP ROUTINE
00EB 00          2880  NOP
00EC 00          2881  NOP
00ED FB          2885  EI
00EE E9          2890  PCHL
00EF E9          2900  PCHL
```

```
00F0 E9     2906   PCHL
00F1        2918 * PROGRAMMER VERIFICATION ROUTINE
00F1        2919 * PART 3
00F1 3B     2920   DCX SP
00F2 F1     2930   POP 6
00F3 EB     2940   XCHG
00F4        2949 * COMPARE FOR GREATER
00F4 BE     2950   CMP M
00F5 EB     2960   XCHG
00F6 17     2970   RAL
00F7 E6 01  3000   ANI 1
00F9 2F     3010   CMA
00FA 3C     3011   INR A
00FB 85     3015   ADD L
00FC 6F     3020   MOV L, A
00FD 3B     3030   DCX SP
00FE 3B     3040   DCX SP
00FF        3050 * COMPARE FOR LESSER
00FF F1     3055   POP 6
0100 2F     3056   CMA
0101 EB     3058   XCHG
0102 86     3059   ADD M
0103 EB     3060   XCHG
0104 C6 07  3061   ADI A, 1
0106 3F     3065   CMC
0107 17     3070   RAL
0108 E6 01  3090   ANI 1
010A 2F     3100   CMA
010B 3C     3101   INR A
010C 85     3105   ADD L
010D 6F     3110   MOV L, A
010E 03     3130   INX B
010F 13     3140   INX D
0110 78     3150   MOV A, B
0111 E6 04  3180   ANI 4
0113 2F     3190   CMA
0114 3C     3191   INR A
0115 85     3195   ADD L
0116 6F     3200   MOV L, A
0117 E9     3210   PCHL
```


LET'S HEAR FROM YOU

Your letters are the lifeblood for us

We really do appreciate your letters and comments about INTERFACE AGE. We need to know what you think about the publication so that we know in what ways we can make it better.

If you like it, tell us, and if there are things you think could be improved, tell us that, too.

We do read all our mail and we appreciate your feedback.

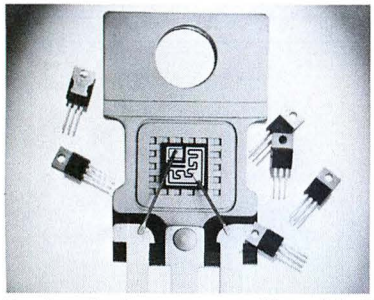
The Editors

New Products

COMPONENTS

Motorola to "Alternate-Source" Popular Plastic Power Transistors

Motorola has supplemented its extensive line of plastic-packaged power transistors with a series of 20 new devices that duplicate the more popular TI and RCA-registered devices. Included in the new offering by Motorola are six Darlington devices and 14 discrete transistors in the 65 Watt power category. These devices are being introduced at prices that are 5 to 10 percent below published competitive prices at the time of introduction.



The introductions represent Motorola's first use of the TO-220 plastic package for silicon power transistors. This is in addition to the plastic cases, 77, 90 and 199 which have housed the company's plastic power line up to now. Customers however can continue to purchase devices with similar electrical specifications in the other Motorola plastic packages.

Motorola plans to extend its use of the TO-220 package to other power device types as quickly as production capability permits.

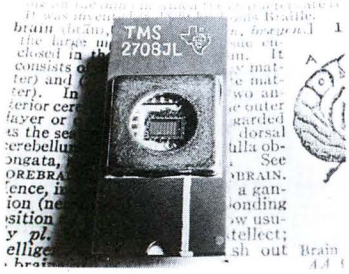
Product availability is now, from the factory and from authorized Motorola distributors.

For further information contact Silicon Power Marketing at (602) 244-4284 or the Technical Information Center, Motorola Semiconductor Products Inc., P.O. Box 20294, Phoenix, AZ 85036.

CIRCLE INQUIRY NO. 90

8,192-Bit UV Light Erasable Programmable Read-Only Memory

The TMS 2708JL, is pin-for-pin compatible with the Intel device with the same part number.



Maximum access and minimum cycle times are 450 nanoseconds. The 184 x 124 mil chip is more than 11 percent smaller than the Intel chip, potentially allowing improved manufacturing productivity.

This smaller chip size is due to TI's unique periphery circuit design techniques which also reduce power consumed by the TMS 2708. Typical power dissipation is less than 450 milliwatts.

The memory circuit is organized as 1,024 words of 8-bit length. It is designed for high-density, fixed-memory applications where fast turnarounds and/or program changes are required.

The device is fabricated using an N-channel silicon-gate technology for high speed and simple interface with MOS and bipolar circuits. All inputs can be driven by Series 74 TTL circuit without external resistors.

The data outputs are three-state for OR-tieing multiple devices on a common bus. A pin-compatible mask programmed ROM, the TMS4700, is available for large volume systems.

The EPROM can be erased by exposing the chip through the transparent lid to high-intensity ultraviolet light at a recommended exposure of 10-watts-seconds per square centimeter.

The TMS2708 is supplied in 24-pin dual-in-line ceramic packages designed for insertion in mounting-hole rows on 600-mil centers and operates from 0°C to 70°C.

Prices for each part are as follows: 1-24, $98.00; 25-99, $73.00; 100+, $64.00. Parts are available now.

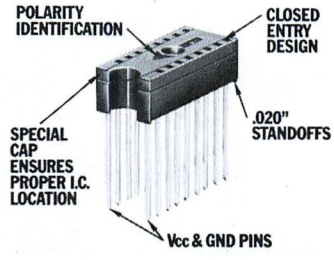
For further information contact Texas Instruments Incorporated, Inquiry Answering Service, P.O. Box 5012, M/S 308 (Attn: TMS 2708), Dallas, TX 75222; (713) 494-5115, Ext. 3281.

CIRCLE INQUIRY NO. 91

16-Pin Discrete Wrap Sockets

A unique 16-pin discrete arrangement saves labor, time, and machine costs by eliminating the need for mounting separate power and ground pins into the board. The new socket contains two additional dummy pins for commitment to Vcc and GND, and also has a specially designed cap that ensures proper insertion and location of 16-pin DIP devices.

Depending on how a particular I.C. socket board is configured, many significant savings can be realized through use of the 16-Pin Discrete. The socket need only be plugged into the board as the power and ground pins are provided as part of the socket unit. A patent has been applied for on the design of the Discrete socket.



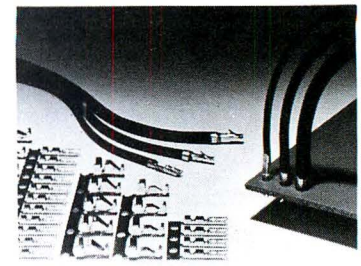
The 16-Pin Discrete has proven so successful in cost savings that Scanbe itself has taken advantage of the new concept by introducing several new 16-pin socket cards and panels, and is passing the savings gained by using the new sockets on to its customers. Typical pricing of the sockets is 39 cents each in lots of 1000.

For further information contact Scanbe Marketing Services, a Division of Zero Mfg. Co., 3445 Fletcher Avenue, El Monte, CA 91731; (213) 579-2300.

CIRCLE INQUIRY NO. 92

Strain Relief Terminal

The 4811 series terminals are used to secure wire leads to a printed circuit board prior to soldering.



Strain relief terminal eliminates leads falling out of the printed circuit board holes during component insertion and handling prior to soldering. The terminal construction allows the actual wire lead to be soldered, and because the terminal fills the printed circuit board hole, *solder voiding is* minimized. Due to the low insertion force, wires with crimped 4811 series terminals can be inserted into the printed circuit boards at any time during assembly and prior to soldering, without causing adjacent components to "pop" out of the printed circuit board due to excess board deflection. After soldering, the insulation crimp creates an effective strain relief eliminating the need for pre-affixed eyelets or griplets on the printed circuit board.

For further information contact Al Maag, Molex Incorporated, 2222 Wellington Court, Lisle, IL60532; (312) 969-4550.

CIRCLE INQUIRY NO. 93

Single-Chip Microprocessor Controller

The MC10801 Microprogram Control Function is a bipolar LSI circuit, a member of the M10800 MECL Processor Family which controls the sequence of the microprogram instructions stored in a processor's control memory. The MC10801 responds to a set of 16 powerful jump and branch instructions. Mnemonic labels for each of the instructions simplify microprogram development by expressing program flow in easy-to-use assembly level language format. While designed for the M10800 Processor Family, the MC10801 is also useful as the microprogram controller in a MECL 10,000 system.

Five 4-bit I/O ports are available on the chip to shuttle control memory address information. Eight 4-bit master-slave registers in the MC10801 hold the current microprogram address, cycle as an index counter for repeats, store op-codes and flag conditions, and nest subroutines in a 4 x 4 push/pop LIFO stack. The MC10801 also contains a "next address" logic section, which, in conjunction with the 16 jump and branch instructions, determines the information, for the next control memory address. Additional internal gates and multiplexers transfer data to and from the registers; three pins on the package convey processor status to the MC10801 for branch decisions. While organized as a 4-bit slice, MC10801s can be connected in parallel to accomodate larger control memory word length.

Compatible with all MECL 10,000 circuits, the MC10801 requires −5.2V and −2V supplies, operates over the temperature range −30°C to +85°C and is housed in a 48-pin Quil* (quad-in-line) package. The price is $50.00 in 100-999 quantities.

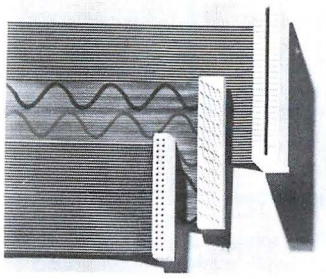
For further information contact Bi-Polar Marketing at (602) 962-2151 or the Technical Information Center, Motorola Semiconductor Products Inc., P.O. Box 20294, Phoenix, AZ 85036.

* Trademark of Motorola Inc.

CIRCLE INQUIRY NO. 94

More Great Jumper Socket Connectors

A faster, easier and less expensive way of implementing standard double-row socket-connector-terminated flat cable systems is an extraordinary family of Great Jumper™ Socket Connectors.



Like all Great Jumpers, these are fully pre-assembled and fully pre-tested flat ribbon cable/connector assemblies. They are available in a wide variety of configurations including several line widths, a choice of conductors, and a selection of opposite end terminations.

Particular to the Great Jumper family is an especially useful doubly-encoded Rainbow cable. This cable is color coded line-by-line on the front, and color-striped in groups of ten on the reverse. This Rainbow cable is intended for use with single-ended Great Jumpers.

The standard cable color in the Great Jumpers family is a distinctive Electric Pink.

Double-ended and daisy-chained (Great Daisy Jumpers) configurations can include not only socket connectors, but solderable printed circuit board connectors and card-edge connectors as well.

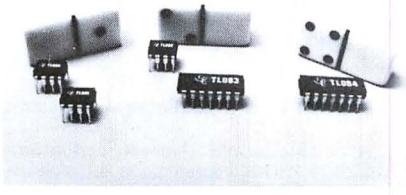
The Great Jumpers Socket Connectors use double rows of contact sockets on .100" centers. Like all great Jumpers connectors, they are molded on during factory assembly and include integral strain relief and line-by-line probeability.

For further information contact A P Products Representatives, who can be located through A P's toll-free Faster and Easier Line, (800) 321-9668.

CIRCLE INQUIRY NO. 95

Full Line of BIFET OP Amps

A full line of five BIFET operational amplifiers, including single, dual and quad circuit types, is available now from Texas Instruments Incorporated.



The op amps incorporate well matched, high voltage JFET and bipolar transistors in a monolithic integrated circuit. They feature high slew rate, low input bias and offset currents, offset voltage selection and a low offset temperature coefficient.

The single type circuits, TL080 and TL081, have offset voltage null capability and the latter requires no frequency compensation. The dual circuits, TL082 and TL083, both include internal frequency compensation. The TL083 also has offset voltage null capability. All four devices complement the previously announced TL084 quadruple op amp.

The TL080, TL081 and TL082 are offered in 8-pin plastic DIP and TO99 metal can packages. The TL083 and TL084 are available in 14-pin plastic DIP.

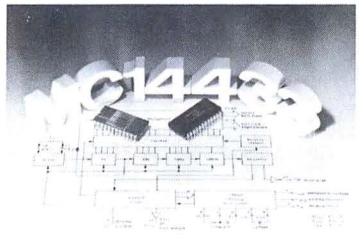
Prices for each part in 100-piece plastic packages for the commercial temperature range (0° to 70°C) are as follows: TL080CP, $1.04; TL081CP, $ .52; TL082CP, $ .91; TL083CN, $1.17; TL084CN, $1.30. All devices are available now through TI distributors.

For further information contact Texas Instruments Incorporated, Inquiry Answering Service, P.O. Box 5012, M/S 308 (Attn: BIFET OP AMPS), Dallas, TX 75222.

CIRCLE INQUIRY NO. 96

Low Cost 3½ Digit CMOS A to D Circuit

A new single-chip CMOS integrated Circuit that requires only two external resistors and two capacitors and a single voltage reference to form a modified, dual-slope, analog-to-digital converter is now available from Motorola. It is designed for DVM/DMM, digital thermometer, digital scale applications, and can be used in MPU systems. The 3½ digit circuit, designated the MC14433, has a multiplexed BCD output format and has an intrinsic full scale range of ±199.9 mV (200 mV reference) or ±1.999 V (2 V reference), with an input impedance of more than 1000 megohms. The MC14433 dissipates very little power, typically 8 mW for ± 5 volt supplies. This unit operates well with both LED and LCD displays.



Other features include: ±0.05% of reading accuracy; Up to 25 conversions per second; Autopolarity; Auto Zero; Standard B-Series outputs; On-chip or external clock; Overange and underange signals. The MC14433 uses the techniques developed by Motorola to put both linear and digital functions on one chip. The linear functions contained are high-performance, low-power operational and differential amplifiers.

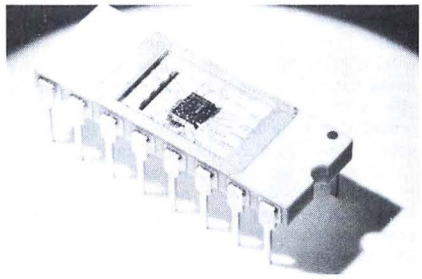
The MC14433 package is a 24-pin dual-in-line in either plastic ("P" suffix) or ceramic ("L" suffix). Pricing in quantities of 100 to 999 is $9.97 for the MC13322P and $14.95 for the MC14433L. Availability is off-the-shelf from the factory or distributor stock.

For further information contact Technical Communications, Motorola Integrated Circuit Division, 3501 Ed Bluestein Blvd., Austin, TX 78721.

CIRCLE INQUIRY NO. 97

8-Bit D/A Converter Features High Speed, Direct Interface to All Logic Families

A new series of high speed multiplying digital-to-analog converters with direct interface to TTL, ECL, HTL, CMOS and PMOS logic families is now available from Signetics.



Designated NE5007/8 and SE5008, the new converters incorporate advanced circuit design that achieves 85-nanosecond settling time and high swing, adjustable threshold logic inputs to provide full noise immunity.

At −15 volts, for example, logic inputs may swing between −10 volts and +18 volts. This makes possible direct interface with CMOS logic, even when the unit is powered from a 5-volt supply.

The 5007/8 units are pin and functionally compatible with monoDAC-08 converters originally introduced by Precision Monolithic Incorporated.

Monotonic multiplying performance in the 5007/8 units is attained over a wide 40 to 1 reference current range. Full-scale current is prematched to ±1 LSB so the need for full-scale trimming is eliminated in most applications. Linearities are as close as 0.19% over the entire operating temperature range, which is −55 degrees C to 125 degrees C for the unit designated SE5008.

Typical applications include converters for servo motors, pen drivers, waveform generators, audio encoders and attenuators, analog meter drivers, programmable power supplies, CRT display drivers, high speed modems and many other applications where low-cost and high speed D/A conversion is required.

Power supply range is from ±4.5 volts to ±18 volts with essentially unchanged performance over the range. The convenient 16-pin plastic DIP package and power consumption of 33 milliwatts with a ±5 volt supply makes the 5007/8 converters ideal for use in portable equipment and for military/aerospace applications.

The 5007/8 units offer dual complementary outputs that permit differential operation, effectively doubling the peak-to-peak output swing. Output compliance is from −10 volts to +18 volts.

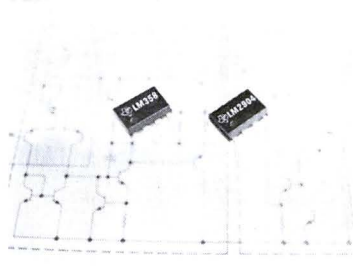
The NE5007/8 and SE5008 converters are available from stock through Signetics and its authorized distributors. Prices in quantities of 100 are $3.45 for NE5007 and $3.95 for NE5008.

For further information contact Signetics, 811 East Arques Avenue, Sunnyvale, CA 94086; (408) 739-7700.

CIRCLE INQUIRY NO. 98

Two New Dual OP Amps

Texas Instruments has announced two new dual operational amplifiers, the LM358 and LM2904. Both are second source to the National devices with the same designations.



The two high-gain, frequency-compensated op amps were designed to operate specifically from a single power supply over a wide range of voltages.

The LM358 is available in 8-pin plastic and ceramic DIP and 8-pin metal can packages. It is characterized for 0°C to 70°C. Prices are 67 cents per part in 100 part quantities for the plastic package. (The LM158 and LM258 offer different operating temperature ranges, −55°C to 125°C and −25°C to 85°C respectively.)

The LM2904 is also available in 8-pin plastic and ceramic DIP and 8-pin metal can packages. It is characterized for −40°C to 85°C operation. Prices are $1.15 each in quantities of 100 for plastic packages.

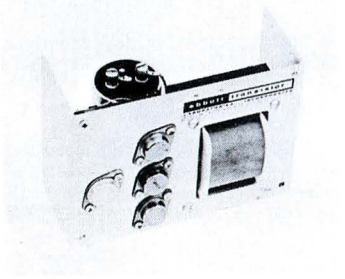
Both devices are available in plastic now at TI distributors. Ceramic and metal can versions will be available soon.

For further information contact Texas Instruments Incorporated, Inquiry Answering Service, P.O. Box 5012, M/S 308, Dallas, TExas 75222 (Attn: LM/358,2904); (214) 238-2955.

EQUIPMENT

Power Module Case Size AAA Triple Output

The "AAA" series of the new NL line provides triple outputs of 5V/3A and either 12V/1.0A ±15V/08A. Standard input is 115 VAC, 47 to 440 Hz with 220 VAC available at no additional cost. Dual primaries are also available. All units feature tight regulation, low ripple and full load operation at 50°C ambient temperature with derating to 40% at 71°C.



Overvoltage protection is standard on 5V outputs and available as an optional feature on the higher voltages. Case size is only 10¼ X 4 x 2½ inches with mounting on three surfaces. High quality components are used throughout with conservative design margins to assure high reliability and long life under worst case operating conditions.

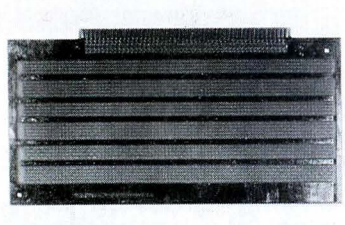
The NL line also includes single, dual and triple output models with power ratings from 15 to 170 watts. Send for Abbott's new 1976-77 Industrial Power Supply Catalog for complete details on this and other lines of power modules. $69.00 (1-24 pieces) normally from stock.

For further information contact Abbott Industrial Products Division, 639 South Glenwood Place, Burbank, CA 91506; (213) 841-2510, Telex 69-6282.

CIRCLE INQUIRY NO. 99

1010 Prototype Board for ALTAIR and IMSAI

The Tarbell Electronics Model 1010 Prototype Board for the ALTAIR* and IMSAI computers is much better than other available boards.



This board accepts up to 33 14-pin IC's, or a mixture of 40-pin, 24-pin, 18-pin, 16-pin, and 14-pin IC's. It is mainly oriented toward soldering point to point, but wire-wrap may also be used. There are three rows for IC's. The IC's or sockets are inserted from the top, then wires are soldered to the adjacent tabs. The tabs are such that even with 40-pin IC's, there are still two holes left over on each pin for wires. A place for a 5-volt regulator is also provided. Edge pins are gold-plated. The price is $28.

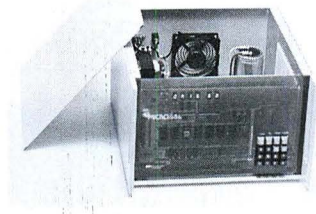
*ALTAIR is a trademark/tradename of MITS, Inc.

For further information contact Tarbell Electronics, 144 Miraleste Drive #106, Miraleste, CA 90732; (213) 538-4251.

CIRCLE INQUIRY NO. 100

Micro-68b™ Microcomputer

The MICRO-68b comes completely assembled with hexkeyboard, 6 digit LED display, 8K RAM, 1K PROM Monitor system, CRT/TTY/Audio Cassette interface.



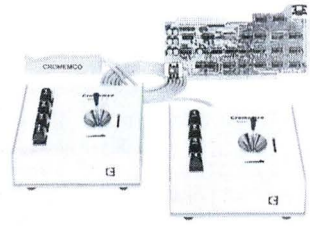
It is housed in a ruggedized aluminum cabinet with a 13 slot exorcisor compatible mother board and 20 amp power supply. The MICRO-68b utilizes the 6800 microprocessor chip set now manufactured by Motorola, AMI, Fairchild, Hitachi and Thompson CSF. The MICRO-68b lists for $1878.00 and is available from stock.

For further information contact Patti Neumann, Electronic Product Associates, Inc., Director of Marketing, 1157 Vega Street, San Diego, CA 92110; (714) 276-8911.

CIRCLE INQUIRY NO. 101

Joystick Console Gives More Flexibility to Microcomputers

A new 2-axis joystick console has significant features not found in existing joysticks. The new joystick is a console rather than merely a simple 2-input device.



It is called a console because it includes a speaker and speaker amplifier built into the same housing with the joystick. The speaker greatly enhances the usefulness of this over other joysticks because it facilitates such uses as sound effects for computer and other games — and provides an easy way to obtain such features as acoustic warnings in other applications.

Another interesting feature of the new console is that it includes four pushbutton switches. These can be used to add much more flexibility to such applications as using the console for cursor positioning on a color graphics terminal. In such a case the switches can select desired colors.

Cromemco also offers peripherals that simplify application of the joystick. One is a fast 7-channel analog-to-digital I/O card. This unit will directly interface ***two*** of the consoles to microcomputers. The I/O is a plug-in card.

Cromemco also offers a color graphics interface known as the TV Dazzler™. This unit allows microcomputers to display memory ***in color*** on an ordinary color TV set.

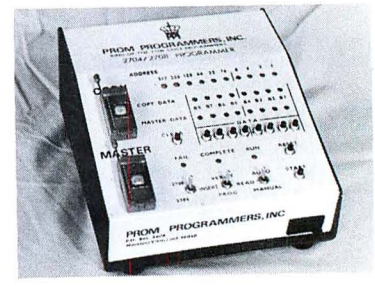
Price of the new console is $65 in kit form or $95 in assembled, ready-to-use form. Delivery is 15-30 days.

For further information contact Mr. Joe McCrate, Cromemco, Inc., 2432 Charleston Rd., Mountain View, CA 94043; (415) 964-7400.

CIRCLE INQUIRY NO. 102

Microprocessor System Boards

The line consists of three boards: central processor board, memory board, and interface board all in the standard 100 x 160 mm EUROCARD FORMAT. Each board is preassembled with the appropriate mini-wrap I.C. sockets, decoupling capacitors, and 64 pin indirect connector plug. The central processor board, for example, comes complete with two 40 pin, four 28 pin, and four 16 pin sockets; fifty mini-wrap terminal pins, eight mini-wrap test point pins, ten decoupling capacitors, and a 64 pin connector plug.



The CPU board is designed to accommodate the microprocessor chip and the minimum memory, clock and interfacing chips. Space is also available for added discrete components and pots enabling the basic system to operate.

The capacity of the CPU can be extended by adding one or more memory boards, each of which are mounted with eight 28 pin sockets designed to accommodate 8K additional memory per board.

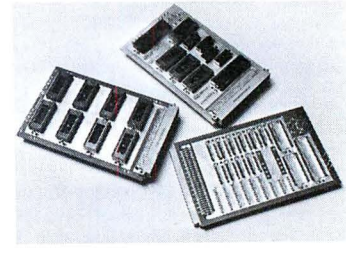
The interface board enables the system to be expanded by providing more comprehensive interfacing and control facilities. In addition to the I.C. sockets, space is also provided for additional discrete components and a crystal permitting the user to construct a crystal controlled clock, which can increase the system's operating speed up to 1 MHZ.

For further information contact Vero Electronics Incorporated, 171 Bridge Road, Hauppauge, N.Y. 11787; (516) 234-0400, TWX 510-227-8890.

CIRCLE INQUIRY NO. 103

2708/2704 PROM Emulating Programmer

The lowest priced, highest performance, 2708/2704 PROM programmer is being claimed by PROM PROGRAMMERS, INC., The King of the Low Cost Programmers. This complete, stand alone, system is priced at $795 and has the capability of both programming and in circuit emulation.



Programming is accomplished using a master PROM or with the binary switches that have been provided for generating or modifying programs within the internal RAM buffer memory.

Interfacing the Master socket of the unit to the in system socket allows the programmer to emulate a 2708 or 2704 PROM. The RAM in the

programmer makes it easy to alter or edit the program being used. Once the data is correct it can be transferred automatically into a 2708 or 2704 PROM by activating the program cycle in the instrument.

Being about the smallest programmer available anywhere the 2½ X 6 X 8 case will easily fit into a briefcase for use in the field or on a desk for use in the factory. Front panel LED's are provided to indicate the state of the address and data lines for both the Master PROM and the Copy. The Programming algorithum is set to Intel's specifications and takes around two minutes to duplicate a Master.

The system includes a General Purpose Interface that enables the PROM to be programmed from a Microprocessor, Mini Computer, etc.

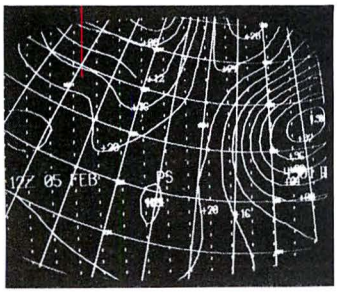
Also available are stand alone units in the same case as the 2708/2704 for the other UV-erasable PROM's, including the 1702A, 5203, and 5204 PROM's. Erase lamp also available. Delivery is from stock to 30 days.

For further information contact PROM Programmers, Inc., 601 Nandell Lane, Los Altos, CA 94022; (415) 948-0450.

CIRCLE INQUIRY NO. 104

Production-Line 1024 X 1024 Digital Raster Graphics Display System Under $20,000

GCT-1024 is a 1024 X 1024 digital raster graphics display system. Its ultra-high resolution takes the "stair-step" appearance out of raster displays to minimize distortion and provide greater density detail.



The complete high-performance, 1024 X 1024 digital graphics display system has been available on a production-run basis and is also priced at under $20,000 (in volume quantities). It includes a unique and proprietary discrete microprocessor that provides instruction times as fast as 150 nanoseconds and 51 mnemonic instructions for increased user programming flexibility.

Coupled with a wide variety of options and accessories such as: a table-mount "joystick" assembly, up to 64-function keyboard units, graphic tablets and cursors, the GCT-1024 is user programmable. Even units with up to 16 gray scales are optionally available.

Applications include functions like Command and Control Process Simulation, Automatic Data Reduction, Computer-Aided Design, Mapping, War Gaming, and Crystallography in such diverse fields as Textiles, Medical, Architecture, Industrial Design, Control Equipment, etc.

Before being officially introduced to the open market the GCT-1024 had already proven its reliability and performance-worthiness in a number of customer installations under stringent-user operating conditions.

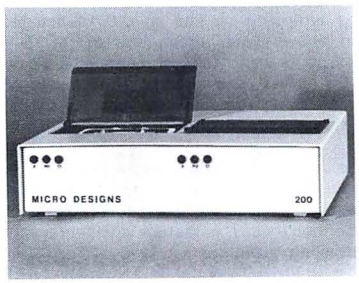
For further information contact William Huber, Genisco Computers, 17805-D Sky Park Circle Drive, Irvine, CA 92714; (714) 556-4916.

CIRCLE INQUIRY NO. 105

Low-Cost Sophisticated Mass Storage for Microcomputers from Micro Designs

Recognizing the need for low-cost versatile mass storage for Altair-type 8080 based microcomputers, Micro Designs is offering two new digital cassette mass storage systems with up to one megabyte capacity.

An integral part of these ready-to-use systems is their complete file management software which allows the user to manipulate both symbolic and binary files with high-level commands.



The Micro Designs Model 100, a compact unit with a single cassette drive, stores one-half megabyte of data. The disk-like format of the data on the tape allows access to any single 128 byte record. The data transfer rate is 1000 bytes per second, and the tape may be searched at a rate exceeding 120 inches per second. The dual transport Model 200 puts one megabyte on line.

Both units come fully assembled, and ready for immediate use. The supplied interface board plugs into the main frame motherboard connector to attach the mass storage unit to the computer. To bring up the operating system, the user loads a cassette, and transfers control to the ROM on the interface board; all further tape operations are automatic. Status lights inform the user of relevant tape conditions, and hardware error detection is provided.

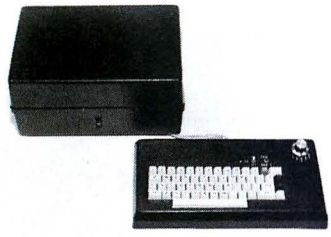
These small table top units sell for $550 (Model 100) and $825 (Model 200). Delivery is 30 days.

For further information contact Micro Designs Inc., 1175 Colusa Ave., Berkeley, CA 94707; (415) 526-7794.

CIRCLE INQUIRY NO. 106

Enclosure Kit for the CT-1024 Terminal

PARSEC ELECTRONICS is now marketing an enclosure kit for the SWTPC CT-1024 Terminal System.



A two enclosure design is used to give the terminal suitable protection while retaining the flexability of a movable keyboard for the user. The two cases are formed from strong, resilient ABS plastic and are color coordinated to blend in with the SWTPC 6800 computer system. The keyboard enclosure has the cutout and mounting pads for the KBD-5 keyboard and a fully enclosed bottom. The main case features mounting pads for the CT-1024 main board, the GT-61 graphics board, the AC-30 cassette interface board and room for their power supplies. The kit includes mounting hardware for all circuit boards, a heatsink for the power supplies and rubber feet for both enclosures.

For further information contact PARSEC, P.O. Box A82327, San Diego, CA 92138.

CIRCLE INQUIRY NO. 107

Digital Interface Unit, DIU-25, Interfaces EMC-25 Interference Analyzers, Electronic Calculators and Computers

Model DIU-25 Digital Interface Unit interfaces the popular Electro-Metrics model EMC-25 Interference Analyzer to the Hewlett Packard HP 9825 and other calculators in the HP 9800 series. In addition, it provides a digital interface to all Hewlett Packard mini-computers and the Digital Equipment Corporation's PDP-8 series of mini-computers.

The new Electro-Metrics Interface Unit decodes digital commands from the calculator or computer, issues corresponding instructions to the Electro-Metrics EMC-25 Interference Analyzer, and then converts the resulting EMC-25 data to appropriate digital format. This data is then fed to the calculator or computer for processing. The result is excellent closed-loop control of EMC-25 Interference Analyzer operation in accordance with the stored program.

An optional Operating Software package, the Electro-Metrics OS-25, permits operation of the EMC-25/DIU-25 combination by personnel ***without prior programming experience***. Development of additional software by the user is simplified by inclusion within the DIU-25 Digital Interface Unit of numerous hard-wired micro-programs.

Of special note in addition to the EMC-25, the new DIU-25 unit also can control key Electro-Metrics accessory devices. These include programmable attenuators, antenna switching units and X-Y plotters.

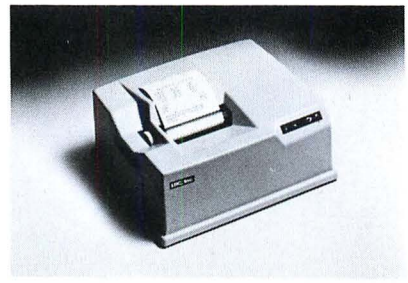
The introductory price is $14,900 and shipments are available in less than 60 days after the date of order.

For further information contact David Cook, Electro-Metrics Division, 100 Church Street, Amsterdam, N.Y. 12010; (518) 843-2600.

CIRCLE INQUIRY NO. 108

Stand-alone Printer

Model 140 is a 40-column stand-alone printer system with complete electronics which utilizes the same field-proven dot matrix impact printing mechanism which LRC developed and introduced in early 1975.



The system includes a proprietary MOS interface chip which includes a 48-character buffer, a character generator, and complete control logic. Through simple external commands, character width and density may be changed at any time, even during printing. Double-width as well as double-high characters may be generated at any time via simple external instructions.

A long list of available options includes; RS-232-C (or current-loop) communications interface, pin-feed platen, high-speed paper feed, label printing (including label stripping), as well as front-feed document printing.

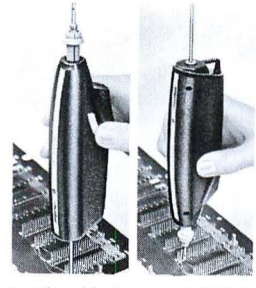
Single-unit price is $495 with substantial discounts for OEM quantities. Deliveries begin in January, 1977.

For further information contact LRC, Inc., Riverton, Wyoming 82501; (307) 856-6524.

CIRCLE INQUIRY NO. 109

Double-ended Tool Speeds Wire Wrapping; Unwrapping

A new double-ended tool, Model P160-9 from Vector Electronic Company, has a wrapping bit on one end and an unwrapping bit on the other for fast, single-tool wrapped-wire terminations. Designed for use either with Vector's battery-powered P160-4R power tool or for manual wrapping, the tool makes gas-tight terminations on 0.025 inch square posts, using 26 to 30 AWG wire. To remove a wire, the tool is simply turned end for end.



The wrapping bit has an 0.070 inch radius which allows wire termination of 0.025 inch square posts on 0.100 inch centers. A bit depth of 0.56 inches permits three wrap levels. Minimum stripping force for six turns of 26 AWG wire is six pounds; three pounds for 30 AWG wire.

The unwrapping bit has a spring-loaded sleeve that retains and ejects the wire after removal, preventing loose wire from falling among the interconnections. This feature saves time wasted in retrieving the wire.

The P160-9 Wrap-N-Unwrap tool is configured for either left-hand or right-hand manual wrapping and unwrapping. The power tool has a right-hand rotation for wrapping with the P160-9 tool; however, by inverting the power tool, left-hand unwrapping is achieved.

The P160-9 Wrap-N-Unwrap tool is priced at $18.40, and the low-cost P160-4R power tool is priced at $45.68. Both are available from stock.

For further information contact Vector Electronic Company, Inc., 12460 Gladstone Avenue, Sylmar CA 91342; (213) 365-9661, TWX 910-496-1539.

CIRCLE INQUIRY NO. 110

Burn-Out-Proof DC Power Supply

Model 244 MOBIL/COMM Power Supply is a compatible, rack-mountable dc power supply addition to the Hickok CB service system.

The Model 244 MOBIL/COMM Power Supply offers features of particular value to service technicians. The fully-adjustable voltage range of 10.5 to 14.5 volts is accurately metered on large 2½" meter with the calibrated standard 13.8 volt setting clearly indicated. Full adjustability and 0.5% regulation permits duplication of actual storage-battery operating conditions such as low-voltage and over-voltage operation.

Continuous-duty three ampere output is protected against short circuits by fold-back current limiting is that during high current-load conditions that may exist in malfunctioning transceivers the power supply will not shut off, but, automatically reduce the current output to a relatively safe level so troubleshooting can continue with current flowing at a reduced level. When the meter switch is in the AMPS position the output current is indicated with 3% accuracy. The fold-back current limiting is self recovering. After the short circuit is removed the unit returns to normal operation, no fuse to replace, no circuit breakers to reset. All overload conditions are indicated by a front-panel OVERLOAD light.

Output connections to the Model 244 are convenient 5-way binding posts.

The Hickok Model 244 MOBIL/COMM Power Supply is available now at Hickok *distributors* for $125.00.

For further information contact Marketing Services Department, Hickok Electrical Instrument Company, 10514 Dupont Avenue, Cleveland, Ohio 44108.

CIRCLE INQUIRY NO. 111

In-Line Digital CB Monitor Provides First Do-It-Yourself Radio Check for 40-Channel CB

The Hickok CB Monitor 38 is a compact, digital-readout CB Monitor which provides display of frequency, power, and SWR.

This is a CB-user version of the precision communication service instruments produced by

Hickok for CB technicians. Designed to be installed in-line between the transceiver and antenna the compact package provides 6-digit frequency readout accurate to 10 parts per million (.001%), 3-digit power output measurement of 1.0 watt to 10.0 watts or if modified 1 watt to 100 watts, accurate to 5%, and 4-digit SWR readings of 1.00:1 to 10.00:1.

The attractively styled package is about the size of a small mobile CB transceiver 6"w x 2¼"h x 7"d, and comes complete with all mounting hardware. The CB Monitor 38 can be operated from standard line voltage of 105 to 125 Vac or a 12 volt car battery for mobile application.

In normal operation with either AM or SSB transceivers the CB Monitor 38 will provide continuous digital readout of the frequency output of the transmitter, the power output of the final stages of the transmitter and the functional condition of the antenna system and transmission cable. The CB operator can read the operational condition of his communications system and can take corrective action for maximum effectiveness.

The CB Monitor 38 is available now through Hickok distributors. Suggested retail price is $279.00.

For further information contact Marketing Services Department, Hickok Electrical Instrument Company, 10514 Dupont Avenue, Cleveland, Ohio 44108.

CIRCLE INQUIRY NO. 112

SOFTWARE

Mupro 8080 Software Programs Now Available for License/Purchase

The BSAL-80 Block Structured Assembly Language programs for use with the 8080 family of microprocessors is now available on a Program License Charge Basis. The programs are fully compatible with muPro-80 and other 8080 Development Systems.

The BSAL-80 Assembler with Relocating/Linking Loader allows 8080 users to write programs in high-level language syntax while retaining the flexibility, program size and execution speed of assembly language. The ability to utilize any of the 8080 machine instructions and to control the use of the architectural features of the 8080 is maintained.

Object code produced by BSAL-80 is in relocatable form, allowing true modular software design and program implementation. The BSAL-80 Relocating/Linking Loader combines individual program modules produced by the BSAL-80 assembler into one executable object module. The linking loader permits the programmer to specify program starting addresses, data starting addresses, module loading sequence and address space for data variables.

The text editor provides a powerful text generation and editing capability. Text manipulation is facilitated by automatic line numbering and a unique command set including ADD, DELETE, LIST, FIND, REPLACE, MODIFY, GATHER, KEEP and SET commands.

The BSAL-80 assembler and loader are provided in 8080 resident or Standard Fortran IV Cross Assembler versions.

Programs are available in paper tape or floppy diskette media. The programs are also provided free of charge with the purchase of the μPro-80, μPro-80E or μPro-80ED Hardware/Software Development Systems.

BSAL-80 Assembler with BSAL-80 Relocating/Linking Loader is $975. The BSAL-80 Text Editor is $350. The price for both programs purchased together is $1,250. The Standard Fortran IV Crossassembler version is $1,250.00 Delivery is immediate.

For further information contact Jim Moon, muPro Inc., 424 Oakmead Parkway, Sunnyvale, CA 94086; (408) 737-0500.

CIRCLE INQUIRY NO. 113

AMDSAM 2900 Software

A software system to generate microinstructions for use with the Am2900 — AMDASM — system has been developed by Advanced Micro Devices and is now available worldwide through Computer Science Corporation's Infonet Service.

Documentation now available explains how to interact with AMDASM to microprogram all control signals and memory to reduce prototype and develop time for 2900 systems. With the use of variable instruction set microprogramming, field service and enhancements to systems are easy as they require only additional data in the microprogram memory. Automatic documentation for program changes as well as training, documentation and demonstrations of this assembler reduce lead time and cost of writing microcode.

This microprogram assembler provides software assistance and documentation for writing and modifying microprograms and generating tapes for PROM programmers. AMDASM includes a framework for a common language, automatic accounting information and billing control. Budget limits, character rate option and batch rates make AMDASM cost competitive with in-house versions.

AMDASM is on the time-sharing service of Computer Science Corporation, one of the world's largest software companies. It is accessible from a standard time sharing terminal from most major cities.

For further information contact E. Sopkin, Advanced Micro Devices, Inc., 901 Thompson Place, Sunnyvale, CA 94086; (408) 732-2400.

CIRCLE INQUIRY NO. 114

Fun and Games Programs Offered with TI SR-52 Calculator Purchase

A library of 20 entertaining and challenging games of skill and chance is being added to the prerecorded programs Texas Instruments Incorporated offers to users of the SR-52 magnetic card programmable calculator.

They will be given (at no cost) to SR-52 purchasers between November 15 and January 15. The calculator sells for $299.95. After January 15, 1977, the new program library will be available to others at a suggested retail price of $29.95.

Several games in the library require players to solve puzzles using numbers; others simulate sea battles between two ships, hitting a target with mortar fire, parachute jumping, and landing on Mars. There are also games that allow users to play basketball, football, and baseball.

CODE BREAKER — One game, called Code Breaker, challenges the player to determine which one of the over 3,000 possible four-digit numbers the calculator has generated randomly. The number will not include the same digit twice, nor will zeros be used.

As the player keys in his guess of what the number is, the program and calculator display tells him two things. First, he learns how many of his four digits are in the number he seeks, and are also in the correct position. He also finds out the number of digits which are right but in the wrong position.

For example, a player may be seeking a number like 8954 which the calculator has generated. If his guess is 7685, he discovers that he has two of the digits correct, but both are in the wrong position. If his guess is 8459, he learns he has identified all four digits of the number he seeks, and that two are in the right position and two are not.

While it may seem difficult, players can gain proficiency and learn to break the four-digit code rapidly. If, for example, he is trying to identify 8954, and he keys in 1362, he learns he has not guessed any digits correctly so he can disregard those four digits in his later guesses.

STAR BUSTER — A more difficult game in the library is called Star Buster. It uses ones to represent stars and zeros to signify black holes in a simulated "universe." They are arranged in a three by three matrix and the game begins with one star in the center of the matrix and black holes in each of the other eight surrounding points. The object is to shoot at a single star to create new ones.

The problem is that as a player shoots a star, the space it occupied in the matrix becomes a black hole, but new stars are created in adjacent spaces. Shooting a star in the center of the matrix, for example, creates four new stars in spaces above, below and to the left and right of the center space, which then becomes a black hole. The game ends when a player creates eight stars surrounding one balck hole in the center of the matrix.

The game gets complicated because a shot creates two, three, or four stars or black holes depending upon their location within the matrix. A shot at a star, therefore may create new ones in desirable locations, but also may substitute a black hole in a space where the player doesn't want one.

BASKETBALL — The basketball game allows two players to alternate on offense and defense as in a real game, and keeps a running score in the calculator display.

To simulate a play, the player on offense confidentially keys in a number from one to five. After the other player does the same thing, the program determines the play's effectiveness based upon the closeness of the two numbers selected by the players.

For example, "ball" control may switch to the defense if both players key in the number three. If it is not, probability again determines which of the two players takes over on offense.

Based upon a certain number of plays, the calculator will signal the end of the game by causing the scoring information in the display to flash.

SEA BATTLE GAMES — One of the sea battle games is called Phantom Ship. It simulates a ship within a 100 unit X-Y matrix. After a player fires his first shot, the program tells him how many units — but not the direction — the shot missed.

The player can shoot again, but the program will move the ship five units in any direction before he does. Shots must be within five units to score a minor hit, and five of those disable that ship and replace it with a new one at another location. A direct hit is required to sink the ship.

To obtain the games library, those who purchase an SR-52 between November 15, 1976, and January 15, 1977, return a customer information card enclosed in each SR-52 box to "SR-52 Games Library, P.O. Box 1210, Richardson, TX 75080. The offer is good only in the United States and is void where prohibited by law.

For further information contact Texas Instruments Incorporated, Inquiry Answering Service, P.O. Box 5012, M/S 308 (Attn: SR-52 Games), Dallas, TX 75222; (214) 238-2154.

CIRCLE INQUIRY NO. 115

Calculator Programs for Surveyors

A library of prerecorded surveying programs for use with its SR-52 handheld, card programmable calculator has been introduced by Texas Instruments Incorporated.

The flexible, professional programs give the engineer or surveyor advanced calculating power and fast answers both in the field and office.

Included in the library are programs for traverse, closure, balance, vertical and horizontal curve design, EDM and stadia reductions, intersections, earthwork and burrow pit volumes, triangle and curve solutions.

Along with the prerecorded program cards, purchasers of the library receive a program manual with complete user instructions and printed listings of each program and examples of typical problems.

The library has a suggested retail price of $44.95. It is the latest addition to TI's series of SR-52 libraries, including those for mathematics, finance, electrical engineering, statistics, aviation, marine navigation, and basic applications.

Users of these programs load data into the calculator memory from a magnetic card smaller than a stick of chewing gum. Then variable data is keyed in for fast solutions to specific problems.

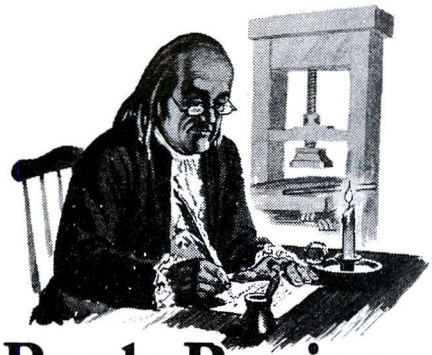
The SR-52 calculator has a suggested retail price of $299.95.

The programs are designed for use by surveyors, civil engineers, highway design engineers, state and municipal engineers and consultants. They provide these professionals with a calculating capability not available before in this price range.

To enhance its capability for field use a 12 volt chamber adapter (DC9100) is also available as an optional accessory for the SR-52, allowing operation from the cigarette lighter in most automobiles.

For further information contact Texas Instruments Incorporated, Inquiry Answering Service, P.O. Box 5012, M/S 308 (Attn: SR-52 Surveyor Pak), Dallas, TX 75222; (214) 238-2154.

CIRCLE INQUIRY NO. 116



Book Review

BASIC SOFTWARE LIBRARY

by Rodger W. Brown

Scientific Research Institute
1712-I Farmington Court
Crofton, MD 21114

VOL I — $24.95
VOL II — $24.95
VOL III — $39.95

- Three volumes consisting of 682 pages
- Includes 102 application programs in BASIC

BASIC APPLICATION SOFTWARE BOOKS INCLUDE BUSINESS PROGRAMS

Rodger W. Brown of SRI has published three volumes of a planned series of 7 BASIC application program volumes entitled BASIC SOFTWARE LIBRARY. The three volumes published to date include 102 BASIC application programs consisting of games, pictures, business, personal bookkeeping, mathematics, engineering, plotting, statistics, and advanced business programs.

The BASIC programs contained in these three volumes obviously represent the culmination of a very large project. This compilation is a versatile and complete BASIC Software Library that will be of use to a large number of diverse individuals. The programs presented in these volumes when combined in an end user system will represent a very powerful software library bank. Such a work as this has been attempted in the past in such areas as cookbooks, electronic product source books, mathematical tables and even computer games. But to date such a collection as this has yet to be offered to the average individual. The word "attempted" was used as no work is ever considered complete by everyone regardless of its thoroughness.

The BASIC programs presented in these three volumes apparently were chosen for their uniqueness and general usefulness. There should be at least several programs in this three volume set that will be of use to every type of individual whether he is a novice or professional programmer. The software programs are written so that little or no computer programming experience is required to run any of the programs. All programs are written in a standard subset of the Dartmouth BASIC Language. Each of the BASIC source code programs have been successfully executed and run on a variety of large, small and even on several of the Microcomputers.

The entire source code is presented as well as a short narrative page which defines the program, tells who might be interested in using it, a brief set of instructions or how to get them via a printout. Limitations in the program are noted and in some cases program examples. In the limitations section the storage length in K Bytes is given so the prospective user will know how much memory to allow for the application program. Where possible the amount of memory space required for full execution is given for each of the programs. This program execution space is independent of the space already occupied by your BASIC compiler or Interpreter. The programs are divided into three volumes each of which is broken down into specific subject sections.

VOLUME 1

Volume One contains 48 BASIC programs subdivided into two sections consisting of Business & Personal Bookkeeping Programs and Games &

Picture programs.

Section 1 contains 20 business type programs. These programs will be of interest to individuals who have businesses, play the stock market, balance their own checkbooks, do installment buying, figure taxes, etc.

Section 2 is the lighter side of the library as it contains 16 games and 12 picture programs. No computer software library is complete without some fun. Among the games presented in this section is one called Checkers. The game is rather long, but it is virtually machine independent, as it does not use overlay techniques nor use files. Most of the other games included here are as exciting as this version of Checkers. Apparently each program was chosen so as not to mimic others that the reader may have seen or used. The pictures are as unusual in their own way as are the games. Most of the pictures are spread over several pages to show details of the particular picture. The picture programs are very simple and it is an ideal place for the novice to start learning about programming.

VOLUME 2

Volume Two contains 46 BASIC programs subdivided into two sections (3 & 4) consisting of Math & Engineering programs and Plotting & Statistics programs.

Section 3 contains 23 general usage Math & Engineering programs. Some of these programs will be of use to high school students, professional people, engineers, astronomers, private airplane pilots, etc. Most of these programs are very technical but they can perform everyday calculations quickly and are extremely simple to use.

Section 4 contains 5 direct Plotting programs and 18 Statistical programs. These programs can be readily utilized by a number of people in widely different disciplines from fishermen to statisticians. The date gathered may be from a poll, a census, a test sample or even the number of fish caught on various days. The stat programs will be of invaluable aid to anyone who gathers data of any kind. The plotting routines will be of use to most of the people who use business programs from section 1, Math & Engineering programs from section 3, and Stat programs from this section. The plotting is done on any standard teletype or terminal and does not require a special plotter or plotting terminal.

Appendix A - Volume 2 also includes an Appendix where the BASIC language grammar is defined. All of the BASIC statements used throughout the BASIC programs are defined here. Each statement is explained sufficiently well to enable one unfamiliar with this subsect of BASIC to modify any necessary statements so that the program or programs will compile and/or execute with the BASIC *Compiler or Interpreter available with* their particular computer. Most of the BASIC Compilers available today that require more than 10K Bytes of storage will execute all of the programs presented in these three volumes with the possible exception of a few of the games and the program "Variable." Multiple line statements are not used in most of the programs and only a few programs are string manipulations extensively. A few of the programs may require more line storage than is available on some of the small Microcomputer systems; these longer programs will not be executable because of the limited amount of memory. However, most of the programs will execute in 10K Bytes of memory or less, thereby making most of the programs in this Library executable in virtually any BASIC speaking computer without any required modifications.

VOLUME 3

Volume three contains 8 BASIC programs on advanced business programs consisting of Invoice Billing & Accounts Receivable program, Inventory Control program, Payroll with deduction program, Capital Investment-Risk Analysis program, Resource Scheduling program, Transportation cost & scheduling assignment program, Stock Value program and a Bond switch Decision program.

Have you told a friend about INTERFACE AGE?

SOFTWARE SECTION

By Robert A. Stevens
Software Editor

FILLING THE SOFTWARE LANGUAGE GAP

INTERFACE AGE takes another step forward in filling the Microcomputer software void. This issue includes 4 articles on software which include 3 major software development program source listings and one BASIC language math function software source listing.

Two BASIC interpreters, one on Tiny BASIC and one on standard BASIC provide conversational language programming capabilities for the SC/MP and 8080 based Microcomputers. National's NIBL TBX Interpreter by Mark Alexander provides conversational language program development for the SC/MP while the LLL's BASIC Interpreter by John Dickinson, Jerry Barber and others provides conversational language program development for the 8080. In addition, both of these highly commentated source listings provide a detailed insight into language programming for those of you interested in BASIC language development. The complete NIBL assembly listing is published in this issue while part two of the LLL8080 BASIC Interpreter-BASIC assembly listing without Floating Point is published in this issue. Assembly listings for the other two parts of the LLL BASIC Interpreter will be published in the next two issues of INTERFACE AGE.

A new resident 8080 software package called CONSOL developed by Processor Technology provides a resident software operating system for their new single PCB intelligent Microcomputer Terminal called SOL Terminal Computer. The complete CONSOL assembly listing is published in this issue.

BEST ARTICLE OF THE YEAR AWARD

INTERFACE AGE will bestow an Honorary Award of $500 value in products advertised in INTERFACE AGE to the author of the best non-commercial Microcomputer article published during the year ending November 1977. The best article of the year award will be picked from the group of the best article of the month awards. Like the best article of the month award, only individuals are eligible for this yearly honorarium. The yearly award is in addition to the monthly award and honorarium given on the page count basis. The yearly award will be judged by the editors of INTERFACE AGE and will be announced in February 1978 issue of INTERFACE AGE.

BEST ARTICLE OF THE MONTH AWARD UP-DATE

Starting in last month's issue, INTERFACE AGE will bestow an Honorary Award of $100.00 to the author of the best non-commercial Microcomputer article of the month. Only individuals are eligible for this monthly honorarium. This monthly award is in addition to the honorarium given on the page count basis. Microcomputer articles may be on hardware, software or a combination hardware-software and will be judged by the INTERFACE AGE readership.

All valid bingo vote cards must be postmarked prior to 12:00 P.M. of the last day of the month ***following*** the issue date of the related magazine.

DON'T FORGET TO VOTE

Help INTERFACE AGE determine the type of articles you want to see published in the future by casting your vote of 10 points for the article or articles (by vote splitting) you liked best. Feedback will provide encouragement to authors and will help make the INTERFACE AGE the Microcomputer magazine of the industry. See the December 1976 issue for bingo card voting details.

INTERFACE AGE WILL PAY UP TO $50/PAGE FOR SOFTWARE

INTERFACE AGE is continually soliciting original unpublished quality documented highly commentated source/object code software listings and software technical articles for publishing in the INTERFACE AGE. Manuscript text must be typed double spaced with wide margins. Figures, tables, flow diagrams and charts must be numbered and submitted on separate sheets of white bond paper (Send original copy only). Program listings must be printed on white clean paper using a new black ink ribbon, and please, if possible, supply a punched paper tape assembly (source-object) code listing + source code listing + object code dump with your hard copy. Be sure to record your name, company and office and home telephone numbers on all material submitted to the software editor. Also include statement in cover letter allowing INTERFACE AGE and the Microcomputer Software Depository to publish and distribute copies of your software program. Include a prepaid postage stamped envelope with your return address only if you want your manuscript returned, in the event that the submitted article is not accepted for publication.

Articles accepted and published will receive an honorary recognition award. Honorariums are based

upon technical content, manuscript preparation and subject suitability for publication in INTERFACE AGE. Honorariums range from $15.00 to $50.00 per typeset magazine page. In addition, starting with this issue, the best article of the month submitted will receive a $100 bonus. INTERFACE AGE's readership will determine by vote which is the best article. (See best article of the month award). All software submitted to INTERFACE AGE will be deposited in the Microcomputer Software Depository (MDS) for low cost distribution.

Address all software correspondence to
R.A. Stevens
Software Editor c/o INTERFACE AGE Magazine
2361 E. Foothill Blvd.
Pasadena, CA 91107 or call (213) 449-1655.

INTERFACE AGE SOFTWARE SHOPPING LIST

Now that INTERFACE AGE has expanded the Microcomputer software coverage and developed a large appetite for good software, your programs and application software is badly needed to quench this enlarged software appetite. This software shopping list includes the following:

- Microcomputer Development Software
- Short Software Routines
- Small Business Programs
- Hardware Control Programs
- Personal Bookkeeping Programs
- Personal Investment Programs
- Stock Market programs
- Small Fry Educational Programs
- Software Communications Protocol Programs
- Off-Line Software Mass Storage Format Control Programs
- Game Programs
- Math Plotting Programs
- Statistics Programs
- Engineering Programs
- You Name It Programs

SECOND CALL FOR INFORMATION ON BASIC PROGRAMMING LANGUAGES

INTERFACE AGE is conducting a survey on the characteristics and programming power of Microcomputer BASIC conversational programming languages. This survey includes Tiny BASIC (TB), Tiny BASIC Extended (TBX), Standard BASIC (SB), Standard BASIC Extended (SBX) and Business BASIC (BB) languages. One of the main objectives of this survey is to highlight the correlation between BASIC languages in order to provide insight for running a BASIC application program on any of the different BASIC languages. If you have developed, helped develop or modified any BASIC type of programming language for any Microcomputer please contact or send hard copy of grammar, users manual and any supporting documentation to:

Robert A. Stevens
Software Editor
INTERFACE AGE
2361 E. Foothill Blvd.
Pasadena, Calif., 91107
Or Call (213) 449-1655

Please include your home and work telephone numbers (for coordination) with all correspondence.

INEXPENSIVE MICROCOMPUTER SOFTWARE

The Microcomputer Software Depository (MSD) will act as repository for source and object code tapes. Programmers wishing to contribute programs to the public domain but who do not want to bother with distribution, may do so by forwarding appropriate documentation including short descriptive write-up and punch paper tape copy of program if possible or cassette copy to MSD. There is no membership fee for access to the public domain paper tapes (PDT) from MSD.

Anyone may obtain copies of these PDT software packages by prepaying a small fee with the order to cover duplication, postage and handling cost. Prices will be listed periodically in INTERFACE AGE. Typical cost for a short program will be approximately $2.00 ($2.00/ounce) + tax, postage and handling. As a convenience MDS will also provide punched paper tape copies of vendor supplied software packages (VSP) that will be sold at vendor suggested sale prices. For a current copy of the available software from the Microcomputer Software Depository (MDS) send a check for $1.00 with a prestamped return envelope to MSD.

Support MSD to build a software library by sending

copies of your documented software programs including short description, flow diagrams and punched paper tape source code and object listings if possible or cassette tape copy for low cost distribution to the following address

Microcomputer Software Depository
2361 E. Foothill Blvd.
Pasadena, CA 91107
(213) 449-0616

MICROCOMPUTER SOFTWARE DEPOSITORY PROGRAM LISTING

THE FOLLOWING LISTS SOFTWARE AVAILABLE FROM MSD ON A PREPAID BASIS ONLY. THE TOTAL COST OF EACH PACKAGE IS THE SUM OF THE BASIC PRICE + CALIFORNIA SALES TAX, IF APPLICABLE, + POSTAGE AND HANDLING COST. FOREIGN SUBSCRIBERS PLEASE NOTE THE DIFFERENT MAILING COST FOR POSTAGE OUTSIDE USA. ADDRESS ALL INQUIRIES TO

MICROCOMPUTER SOFTWARE DEPOSITORY
2361 E. FOOTHILL BLVD.
PASADENA, CALIF., 91107
OR CALL (213) 449-0616

MICROCOMPUTER SOFTWARE DEPOSITORY (MSD) PROGRAMS DATE JAN. 1977

PROGRAM MEDIA
PTAC PAPER TAPE ASSEMBLY CODE
PTSC PAPER TAPE SOURCE CODE
PTOC PAPER TAPE OBJECT CODE
PTBC PAPER TAPE BASIC CODE
PTAL PAPER TAPE ASSEMBLY LISTING
PTSL PAPER TAPE SOURCE LISTING
PTOL PAPER TAPE OBJECT LISTING
PTOD PAPER TAPE OBJECT DUMP
PTBL PAPER TAPE BASIC LISTING
CTAL CASSETTE TAPE ASSEMBLY LISTING
CTSL CASSETTE TAPE SOURCE LISTING
CTOL CASSETTE TAPE OBJECT LISTING
CTOD CASSETTE TAPE OBJECT DUMP
CTBC CASSETTE TAPE BASIC CODE
CTBL CASSETTE TAPE BASIC LISTING
HCAC XEROX HARD COPY OF ASSEMBLY CODE
HCSC XEROX HARD COPY OF SOURCE CODE
HCOC XEROX HARD COPY OF OBJECT CODE
HCBC XEROX HARD COPY OF BASIC CODE
HCAL XEROX HARD COPY OF ASSEMBLY LISTING
HCSL XEROX HARD COPY OF SOURCE LISTING
HCOL XEROX HARD COPY OF OBJECT LISTING
HCOD XEROX HARD COPY OF OBJECT DUMP
HCBL XEROX HARD COPY OF BASIC LISTING
TEXT XEROX HARD COPY OF PRINTED TEXT
PTTL PAPER TAPE TEXT LISTING
CTTL CASSETTE TAPE TEXT LISTING
MAN MANUAL

SUFFIX C= HAND ASSEMBLED CODE
SUFFIX L= COMPUTER FORMATED LISTING
SUFFIX D= CODE DUMP IN OCTAL OR HEX

NOTES
* CALIF. SALES TAX REQUIRED FROM RESIDENCE OF CALIF.
: USA POSTAGE + HANDLING OR THIRD CLASS USA POSTAGE + HANDLING OR SURFACE RATE FOREIGN POSTAGE @ THREE TIMES THIRD CLASS USA POSTAGE RATE (STANDARD) OR SURFACE RATE FOREIGN POSTAGE @ FIVE TIMES USA POSTAGE RATE (ALTERNATE)
< NEW PROGRAM LISTING
% VENDOR SOFTWARE PACKAGE - ALL OTHER PROGRAMS ARE PUBLIC DOMAIN SOFTWARE WITH THE ONLY RESTRICTION THAT THESE PROGRAMS ARE NOT TO BE SOLD IN ANY FORM AT ANY PRICE.

DEFINITIONS:

ASSEMBLY LISTING: COMPUTER ASSEMBLED SOFTWARE PROGRAM LISTING THAT INCLUDES SYMBOLIC ASSEMBLY LANGUAGE SOURCE CODED INSTRUCTIONS WITH COMMENTS PLUS EQUIVALENT MACHINE LANGUAGE OBJECT CODED INSTRUCTINS AND MEMORY ADDRESS ASSIGNMENTS FOR EACH INSTRUCTION (SOURCE + OBJECT).

ASSEMBLY CODE: SAME CONTENT AS ASSEMBLY LISTING BUT HAND ASSEMBLED.

SOURCE LISTING: SOFTWARE PROGRAM LISTING RESULTING FROM COMPUTER SOFTWARE CONTROLLED ASSEMBLY PROCESS THAT INCLUDES ASSEMBLY LANGUAGE SOURCE CODED INSTKUCTIONS WITH COMMENTS. SOMETIMES, LINE STATEMENT NUMBERS ARE INCLUDED FOR EACH INSTRUCTION.

SOURCE CODE: SAME CONTENT AS SOURCE LISTING BUT HAND ASSEMBLED.

OBJECT LISTING: SOFTWARE PROGRAM LISTING RESULTING FROM COMPUTER SOFTWARE CONTROLLED ASSEMBLY PROCESS THAT ONLY INCLUDES MACHINE READABLE OBJECT CODED INSTRUCTIONS AND MEMORY ADDRESS ASSIGNMENTS.

OBJECT CODE SAME CONTENT AS OBJECT LISTING BUT HAND ASSEMBLED.

HARD COPY: XEROX OR PRINTED COPY.

CODE: HAND ASSEMBLED CODE (SOURCE, OBJECT, OR ASSEMBLY CODE).

LISTING: COMPUTER FORMATED LISTING.

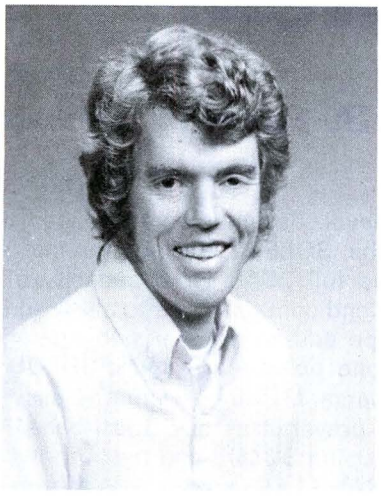
DUMP: COMPUTER MEMORY DUMP.

MSD PROGRAMS

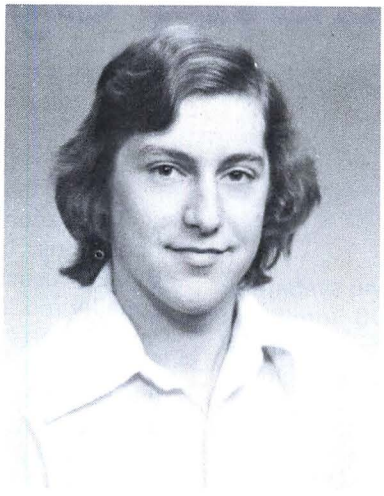
CPU TYPE	SYMBOLIC NAME	DESCRIPTIVE NAME	MSD # & MEDIA	R E V #	PRICE IN $ + CALIF. TAX(*) + USA POSTAGE(:)
6502	APPLECD	6502 APPLE COMPUTER DISASSEMBLER BY ALLEN BAUM & STEPHEN WOZNIAK-INTERFACE AGE, SEPT. 1976, VOL.1,#10.	1-TEXT 1-HCAL		1.00+0.06+0.50 INC. WITH TEXT
8080	LPTIIHF	LOAD 8080 PAPER TAPE IN INTEL HEX FORMAT BY BURT HASHIZUME-INTERFACE AGE, OCT. 1976, VOL.1,#11.	2-PTAL 2-TEXT 2-HCAL	Ø	2.00+0.12+1.00 1.00+0.06+0.50 INC. WITH TEXT
8080	BFWOA	8080 BINARY FILES WITH OPTIONAL AUTOSTART BY WILLIAM H. JORDAN-INTERFACE AGE, OCT. 1976, VOL.1,#11.	3-PTAL 3-PTOD 3-TEXT 3-HCAL	Ø	2.00+0.12+1.00 INC. WITH PTAL 1.00+0.06+0.50 INC. WITH TEXT
6800	MINOPS	MIN OPERATING SYSTEM BY ED KEITH & DENNIS HESCOX-INTERFACE AGE, OCT. 1976, VOL.1,#11. PTAL+ INCLUDES OPERATING INSTRUCTIONS, PAPER TAPE FORMAT AND SAMPLE RUN	4-PTAL+ 4-PTOD 4-TEXT 4-HCAL	< Ø	2.00+0.12+1.00 INC. WITH PTAL 1.00+0.06+0.50 INC. WITH TEXT
8080	DBBDP	DR. BEATTIE'S BASIC DIET PLANNING BY DR. BEATTIE-INTERFACE AGE, OCT. 1976, VOL.1,#11.	5-TEXT 5-HCSL 5-PTSL	Ø	1.00+0.06+0.50 INC. WITH TEXT 3.00+0.18+1.00
6800	EZMERPS	ECHO 1, ZERO MEMORY, ECHO REVERSE & PRINT SUBROUTINES BY HOWARD BERENBON-INTERFACE AGE, OCT. 1976, VOL.1,#11.	6-PTAL 6-TEXT 6-HCAL	Ø	2.00+0.12+1.00 1.00+0.06+0.50 INC. WITH TEXT
8080	ESP-1	ESP-1 SOFTWARE PACKAGE BY MICHAEL SHRAYER-INTERFACE AGE, OCT. 1976, VOL.1,#11. PTGR IS PAPER TAPE COPY OF GRAMMAR.	7-PTOD % 7-MAN 7-CTOD < 7-MAN 7-PTGR < 7-TEXT	Ø	30.00+1.80+1.50 INC. WITH PTOL 30.00+1.80+1.50 INC. WITH CTOD 5.00+0.30+1.50 INC. WITH PTGR
8080	PTSP-1	PROCESSOR TECHNOLOGY SOFTWARE PACKAGE NO. 1 SUMMARY BY R. A. STEVENS-INTERFACE AGE, OCT. 1976, VOL.1,#11.	8-PTTL % 8-TEXT	Ø	1.00+0.06+0.60 INC. WITH PTTL

8080	FRAMMT	EXHAUSTIVE 8080 RAM MEMORY TEST PROGRAM BY T.E.TRAVIS -INTERFACE AGE, NOV. 1976, VOL.1,#12.	9-PTAL 9-PTOD 9-TEXT 9-HCAL 9-HCOD	0	2.00+0.12+0.50 INC. WITH PTAL 1.00+0.06+0.50 INC. WITH TEXT INC. WITH TEXT
6800	MEMDMP-1	SWTPC 6800 MEMORY DUMP PROGRAM MEMDMP-1 BY GARY KAY-INTERFACE AGE, NOV. 1976, VOL.1,#12.	10-PTAL 10-PTSL 10-PTOD 10-TEXT 10-HCAL	0 < 0 <	2.00+0.12+1.00 2.00+0.12+1.00 INC. WITH PTSL 1.00+0.06+0.75 INC. WITH TEXT
6800	ROBIT-1	SWTPC 6800 ROTATING BIT RAM MEMORY DIAGNOSTIC PROGRAM ROBIT-1 BY GARY KAY-INTERFACE AGE, NOV. 1976, VOL.1,#12.	11-PTAL 11-PTSL 11-PTOD 11-TEXT 11-HCAL	0 < 0 <	2.00+0.12+1.00 2.00+0.12+1.00 INC. WITH PTSL 1.00+0.06+0.75 INC. WITH TEXT
6800	MEMCON-1	SWTPC 6800 SHORT MEMORY ADDRESS CONVERGENCE PROGRAM MEMCON-1 BY GARY KAY-INTERFACE AGE, NOV. 1976, VOL.1,#12.	12-PTAL 12-PTSL 12-PTOD 12-TEXT 12-HCAL	0 < 0 <	2.00+0.12+1.00 2.00+0.12+1.00 INC. WITH PTSL 1.00+0.06+0.75 INC. WITH TEXT
6800	BJIR	BLACKJACK IN BASIC PROGRAM BY ED KEITH & DENNIS HESCOX. THE BJIR PAPER TAPE OBJECT CODE REQUIRES ROBERT UITENWYK'S SWTPC MICROBASIC OPERATING SYSTEM-INTERFACE AGE, NOV. 1976, VOL.1,#12. PTBL+ INCLUDES SAMPLE RUN, INSTRUCTIONS, LIST OF VARIABLES AND LIST OF ROUTINES.	13-PTBL 13-PTBL+ 13-PTOD 13-TEXT 13-HCBL 13-HCOD	0 <	5.00+0.30+1.00 6.00+0.36+1.00 6.00+0.36+0.75 1.00+0.06+0.50 INC. WITH TEXT INC. WITH TEXT
6502	RFPR	REVISED FLOATING POINT ROUTINES FOR 6502* BY ROY RANKIN & STEVE WOZNIAK - INTERFACE AGE, NOV. 1976, VOL.1,#12. NOTE * - ORIGINAL MATH PACKAGE FIRST APPEARED IN DR. DOBB'S JOURNAL, AUG. 1976, VOL.1,#7.	14-PTOD 14-PTAL 14-PTSL 14-TEXT 14-HCAL	1 <	3.00+0.18+0.60 9.00+0.54+1.50 9.00+0.54+1.50 1.00+0.06+0.50 INC. WITH TEXT
6800	HISPDMUP	HIGH SPEED DOUBLE PRECISON MULTIPLICATION SUBROUTINE-HISPDMUP BY PERMISSION AND COURTESY OF MOTOROLA'S M6800 USER GROUP LIBRARY-INTERFACE AGE, NOV. 1976, VOL.1,#12.	15-PTAL 15-TEXT 15-HCAL	0	2.00+0.12+0.50 1.00+0.06+0.50 INC. WITH TEXT
6800	DIV16	REENTRANT 16 BIT DIVIDE SUBROUTINE - DIV16 BY PERMISSION AND COURTESY OF MOTOROLA'S M6800 USER GROUP LIBRARY- INTERFACE AGE, NOV. 1976, VOL.1,#12.	16-PTAL 16-TEXT 16-HCAL	1	2.00+0.12+0.50 1.00+0.06+0.50 INC. WITH TEXT
6800	RENTMUP	REENTRANT DOUBLE PRECISION MULTIPLICATION SUBROUTINE-RENTMUP BY PERMISSION AND COURTESY OF MOTOROLA'S M6800 USER GROUP LIBRARY-INTERFACE AGE, NOV. 1976, VOL.1,#12.	17-PTAL 17-TEXT 17-HCAL	0	2.00+0.12+0.50 1.00+0.06+0.50 INC. WITH TEXT
8080	HOMEC	COMPUTER OR CONTROLLER BY TERRY BENSON, INTEL - INTERFACE AGE, SEPT. 1976, VOL.1,#10.	18-PTAL 18-PTSL 18-TEXT 18-HCAL	0	2.00+0.12+0.50 2.00+0.12+0.50 1.00+0.06+0.50 INC. WITH TEXT
8080	LCST	STARTREK BY LYNN COCHRAN-INTERFACE, JUNE 1976, VOL.1,#7.	19-PTBL 19-TEXT 19-HCBL	0	2.00+0.12+0.75 1.00+0.06+0.50 INC. WITH TEXT
8080	WSPG	WORD SEARCH PUZZLE GENERATOR BY RICHARD S. EDELMAN - INTERFACE, JULY 1976, VOL.1,#8.	20-PTBL 20-TEXT 20-HCBL	0	2.00+0.12+0.75 1.00+0.06+0.50 INC. WITH TEXT
8080	PGBIORHY	BIORHYTHM BY PAUL GREEN - INTERFACE AGE, AUG. 1976, VOL.1,#9.	21-PTBL 21-TEXT 21-HCBL	0	2.00+0.12+0.75 1.00+0.12+0.50 INC. WITH TEXT
8080	WDBIORHY	BIORHYTHMS IN PRACTICE BY WILLIAM L. DONHAN, M.D. - INTERFACE AGE, AUG. 1976, VOL.1,#9.	22-PTBL 22-TEXT 22-HCBL	0	2.00+0.12+0.75 1.00+0.06+0.50 INC. WITH TEXT
8080	RFBJ	BLACKJACK BY RICHARD S. EDELMAN - INTERFACE AGE, AUG. 1976, VOL.1,#9.	23-PTBL 23-TEXT 23-HCBL	0	2.00+0.12+0.75 1.00+0.06+0.50 INC. WITH TEXT
8080	BLUFF	BLUFF BY PHIL FELDMAN & TOM RUGE - INTERFACE AGE, SEPT. 1976, VOL.1,#10.	24-PTBL 24-TEXT 24-HCBL	0	2.00+0.12+0.75 1.00+0.06+0.50 INC. WITH TEXT
6800	RABSIMB	RELATIVE ADDRESS BACK-STEPPER IN MICRO-BASIC BY J. HUFFMAN - INTERFACE AGE, DEC. 1976, VOL.1,#13.	25-PTBL 25-HCBL 25-TEXT	< 0 <	2.00+0.12+0.75 1.00+0.06+0.25 INC. WITH HCBL
6800	TEXT6800	TEXT EDITOR FOR THE SWTPC-6800 BY MARK BORGERSON - INTERFACE AGE, DEC. 1976, VOL.1,#13.	26-PTAL 26-PTOD 26-HCAL	< 0 <	10.00+0.36+2.00 5.00+0.30+1.25 2.00+0.12+1.25
8080	WPATRX	WANG'S PALO ALTO TINY BASIC BY ROGER RAUSKOLB - INTERFACE AGE, DEC. 1976, VOL.1,#13.	27-PTSL 27-PTOD 27-HCAL 27-TEXT 27-HCSL	< 0 <	10.00+0.60+2.00 5.00+0.30+1.50 2.00+0.12+1.50 INC. WITH HCAL 2.00+0.12+1.00
8080	LLLBI	LLL 8080 BASIC INTERPRETER GRAMMAR BY JERRY BARBER & ROYCE ECKARD - SUBMITTED BY F.R. FISHER - INTERFACE AGE, DEC. 1976,VOL.1,#13(PART 1). PART 2 PUBLISHED JAN. 1977, VOL.2,#1. TEXT1 IS PART 1, TEXT2 IS PART 2 AND HCAL2 IS FULL SIZE XEROX COPY OF BASIC ASSEMBLY LISTING WITHOUT FLOATING POINT.	28-TEXT1 28-HCAL2 28-TEXT2	< 0 <	2.00+0.12+1.25 5.00+0.30+2.00 INC. WITH HCAL
SC/MP	NIBL	NIBL-NATIONAL'S TINY BASIC GRAMMAR FOR SC/MP BY PHIL ROYBAL - INTERFACE AGE, DEC. 1976, VOL.1,#13. ASSEMBLY LISTING PUBLISHED JAN. 1977, VOL.2,#1.	29-TEXT 29-HCAL 29-PTSL 29-PTOD	< 0 <	2.00+0.12+1.25 10.00+3.00+2.00 10.00+3.00+2.00 5.00+1.50+1.00
SC/MP	MWBAGELS	BAGELS BY DR. MARVIN WINZINREAD BY PERMISSION & COURTESY OF NATIONAL SEMICONDUCTOR - INTERFACE AGE, DEC. 1976, VOL.1,#13.	30-PTBL	< 0	2.00+0.12+1.00
8080	AMS80	AMSAT 8080 STANDARD DEBUG MONITOR BY RICHARD C ALLEN & JOE KASSER - BYTE # 13, SEPT. 1976, VOL.2,#1. SUBMITTED BY JOE KASSER.	31-PTSL 31-PTOD	< 2 <	10.00+0.60+2.00 3.00+0.18+1.75
6800	BAFCMP	BASIC ALGORITHMS FOR COMMON MATH FUNCTIONS BY MICHAEL P. BURTON - INTERFACE AGE, JAN. 1977, VOL.2,#1.	32-PTBL 32-TEXT	< 1 <	**3.00+0.24+1.00** **1.00+0.06+0.25**

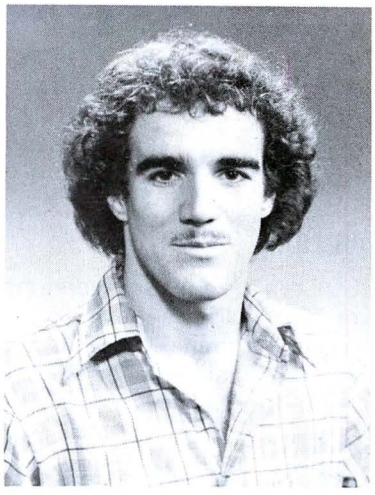
THE 1976 COMPUTER STORE ALL-STARS



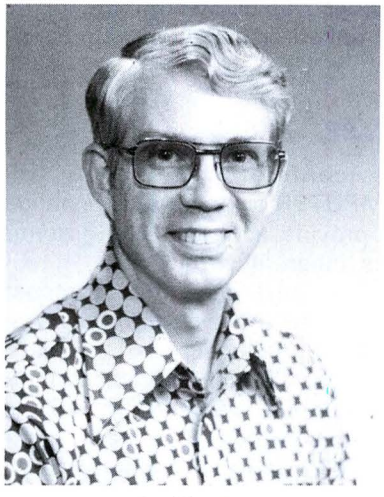
Mike Eusey



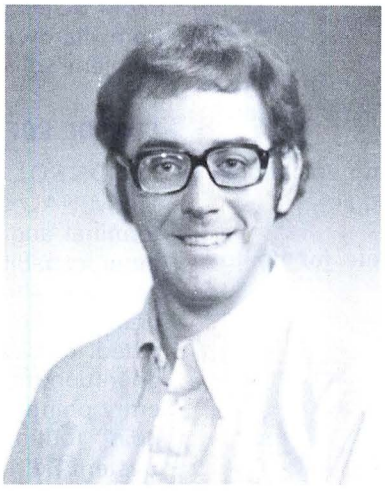
Steve Zook



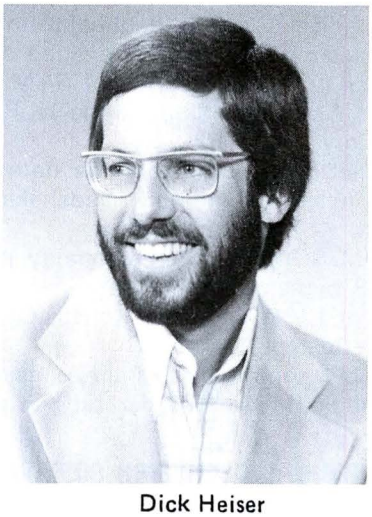
Paul Tripe



John Trotter



Jim Modecki



Dick Heiser

The most experienced computer dealers anywhere

THE COMPUTER STORE

820 Broadway
Santa Monica, Calif. 90401
Phone (213) 451-0713

mits altair™ COMPUTER CENTER

Authorized dealer for:
MITS, Southwest Technical, Polymorphic, Cromemco, Microterm, Oliver, TDL, Sanyo
BY POPULAR DEMAND! NEW HOURS!! TUES – FRI Noon - 8 p.m., SAT 10 a.m. - 6 p.m.
Located 2 blocks North of the Santa Monica Freeway at the Lincoln Blvd. exit

CIRCLE INQUIRY NO. 42

Consol 1K Resident Operating System

By Processor Technology
Emeryville, CA

INTRODUCTION

CONSOL is a resident software operating system for Processor Technology's new single PCB intelligent microcomputer terminal product called SoL Terminal Computer™. The basic SoL Terminal Computer includes the following functional sections;

- 8080A CPU
- 1K bytes of static, low power RAM
- UART controlled RS-232 and 20 ma TTY serial I/O port with multiple baud rates of 75 → 9600 - Switch selectable.
- Video display circuit identical to the VDM-1 sold by Processor Technology. The video display section includes its own 1K bytes of RAM refresh memory. This RAM is in addition to the 1K RW RAM program memory.
- Parallel input-output port for data communications with fully implemented handshaking logic.
- ASCII Keyboard input port.
- PROM/ROM plug-in personality module for up to 2048 bytes of stored program.
- 300 or 1200 baud rate Kansas City Standard Audio cassette tape interface.
- MIB bus compatibility and expansion capability with all Altair/IMSAI/PTC bus plug-in products.

SOL TERMINAL COMPUTER SOFTWARE CONFIGURATIONS

The SoL Terminal Computer™ can be configured by plug-in resident software modules as a stand alone microcomputer or as an intelligent remote editoring terminal.

Basic system operating modes are stored in ROM or PROM on plug-in personality modules with a capacity of up to 2048 words. These modules may be changed in a few seconds to totally reconfigure the system for different applications. Other operating programs, such as BASIC AND FOCAL High level languages, can be loaded automatically into read/write memory (RAM) from cassette tape or floppy disc.

Software control programs for the operation of SoL are designed for three different levels of use. The first level program, CONSOL,™ is contained in 1K of PROM and is designed to allow simple terminal operations. In addition, CONSOL allows direct control of the basic computer functions for entering data to, or examining data in, any memory location, or executing a program stored at a known location in memory.

The second level, designed for advanced terminal operations, is the SOLED™ editing terminal system. SOLED uses the full 2048 word capacity of a personality module and contains code to allow screen, file and cassette tape editing/transmission operations.

The stand-alone operating system, SOLOS,™ turns the SoL into a versatile computer that is easy to use, but every bit as powerful as any 8080-based system available today. Using SOLOS and the built-in cassette interface, BASIC can be loaded in less than a minute following power-on. BASIC programs can be both saved and executed from cassette. The SoL operating under SOLOS, brings true 8080 computer power away from hardware tinkering to direct application and problem solving.

CONSOL SOFTWARE

CONSOL is configured to allow the SoL TERMINAL/COMPUTER to operate as a standard CRT terminal and to provide access to the essential computer capabilities inherent within SoL. The CONSOL software allows self test and small diagnostic programs to be entered to the system memory and executed thus providing verification of correct system operation. In addition, CONSOL contains standardized entry points for all normal I/O operations. These entry point routines are common with each of the SoL System Solftware allowing each personality module in the SoL product line to interface with external programs in an almost identical manner.

A cassette read routine is included in the CONSOL module software allowing SoL System Solftware to be loaded and run in a SoL System with additional memory. SoL System Solftware includes BASIC, FOCAL, a Scientific Calculator and numerous "game" packages including an 8K assembly language version of STARTREK called TREK80.

CONSUL OPERATION

When power is applied to the SoL unit, CONSOL initializes the system ram area, clears the screen, and enters the terminal mode.

In this mode the Sol System acts as a standard CRT terminal sending keyboard data to an output port and displaying received data on the screen. The COMMAND KEYS of the keyboard are not transmitted to the output port but are interpreted as direct internal operation keys. CURSOR MOVEMENT, HOME and CLEAR SCREEN all operate in this manner, while

MODE causes an immediate change in the operation of the SoL Terminal Computer.

When the MODE key is depressed CONSOL issues a prompt (>) and waits for a command line to be entered vie the keyboard. The SoL is now operating as a computer and is ready to accept one of the following commands:

DUmp	Dump memory locations to screen
ENter	Enter data to memory
EXecute	Execute a program in external memory
BAsic	Execute a program located at address zero
TErminal	Return to terminal mode
TLoad	Load program or data from cassette tape
MODE	Press key to start new command line

CONSOL COMMANDS

The seven SoL CONSOL commands are defined in the following:

DUMP <addr> <addr>

The DUMP command displays memory data on the screen in ASCII Hexidecimal representation. As with all SoL commands the command is recognized by the first two characters and up to ten additional characters can be input without an error being forced. Thus, DU; DUST; DUMP; DUMPTHESE would all be recognized as being a DUMP command.

At least one address must follow the command or an error will result. Entering the command DU followed by addr will result in the data at 'addr' being displayed on the screen. If two addresses are defined then all values from the first address to the last address will be displayed. The following example shows the DUMP command with the start and terminating addresses sepcified;

DUMP 0 EF

Up to ten blanks may be inserted between each parameter without forcing an error condition. Errors are flagged by a question mark (?) replacing the character where the error occurred. For example if the DU command were given without an address, the question mark would appear ten spaces to the right of the 'U' character of the dump command.

ENTER addr

The enter command places sequential bytes into memory beginning at the specified address. Data, represented as hexidecimal values, are entered from the keyboard for storage in memory. Values are entered one line at a time with each line terminated by a carriage return or linefeed. The ENTER command function itself is terminated with a slash (/) and the CONSOL operating system returns to the command mode when the slash is encountered.

Data input lines are terminated with a carriage return or line feed. If the terminator is a C/R, CONSOL will erase all characters from the current cursor location to the end of the screen line. In this case, all valid input should be to the left of the cursor. If an error occurred during input the cursor may be moved back to

the left to correct the error.

TLoad <speed>

CONSOL includes routines to read standardized Solftware from cassette tape. This standardized software is recorded on tape with a sixteen byte header that includes NAME, LOAD INFORMATION, FILE TYPE and execute address. CONSOL, because of space limitations, is unable to search for a program or file by name. After receiving the TLOAD command, CONSOL locates the next available data, uses the header information and loads the file to memory.

After loading the data, CONSOL returns to the command mode where the EXEC command can be used to execute the just loaded program. In addition, a return can normally be made to the command mode by pressing the MODE key. Program Space limitations again limited the mechanization of the escape function during the header search, so if the operation software system locks up in this routine the standard SoL restart must be used to escape.

The Audio Cassette Interface electronics within the SoL will record or receive data at either of two standard speeds. TLOAD will accept a parameter to select this speed, 0 being high speed and 1 being low (300 and 1200 bits per second). If no parameter is given, CONSOL will default to high speed operation as all standard SoL-System Solftware is recorded at this speed.

EXecute addr

The execute command is used to run programs located in external memory. CONSOL branches to the external routine in a manner similar to a CALL so that the program can return to the command mode using a standard RET instruction if normal stack operations are used.

BAsic

The BASIC command is provided for executing programs whose starting address is 0. (Such as SoL-BASIC5).

STANDARD I/O ROUTINES

All SoL System personality modules contain similar I/O code for input/output operations. CONSOL, using 1K of memory, has routines for KEYBOARD and SERIAL PORT input as well as SERIAL and VIDEO DISPLAY OUTPUT. Although the same code for SOLOS and SOLED contains expanded functions, the I/O operations appear almost identical when used with external software.

SoL BASIC5 for example performs all I/O using the jump table of the personality modules. Thus, without altering BASIC the user may output to either the serial port or to the display screen. Provision is also made within BASIC to programatically change to any of the four available Input or Output options. CONSOL is of course limited to the two provided.

```
                                ** ALS-8 PROGRAM DEVELOPMENT SYSTEM **

                                     PROCESSOR TECHNOLOGY CORP.
--- CONSOL ---                       6200 HOLLIS STREET
COPYRIGHT 1976                       EMERYVILLE, CALIF. 94608


C000                    0001 *
C000                    0002 *
C000                    0003 *
C000                    0004 *                    *****          *
C000                    0005 *                   *     *         *
C000                    0006 *                   * * *      **   *
C000                    0007 *                   *   *      * *  *
C000                    0008 *                   *****      **   ******
C000                    0009 *
C000                    0010 *                      SYSTEM SOLFTWARE
C000                    0011 *
C000                    0012 *
C000                    0013 *                     <<>>    CONSOL    <<>>
C000                    0014 *
C000                    0015 *
C000                    0016 *     VERSION:  1.0:5
C000                    0017 *     RELEASE: 10.11.76
C000                    0018 *
C000                    0019 *
C000                    0020 *
C000                    0021 *       THIS PROGRAM IS THE MINIMUM SOL OPERATING SYSTEM.
C000                    0022 *  IT PROVIDES ALL NECESSARY DISPLAY ROUTINES ALONG WITH
C000                    0023 *  STANDARDIZED CALLING POINTS FOR INPUT/OUTPUT OPERATIONS.
C000                    0024 *
C000                    0025 *       COMMANDS ARE ALSO AVAILABLE TO ENTER DATA TO MEMORY
C000                    0026 *  AND TO DISPLAY MEMORY DATA ON THE SCREEN.  AN EXECUTE
C000                    0027 *  COMMAND IS PROVIDED TO ALLOW PROGRAM EXCUTION OUTSIDE
C000                    0028 *  OF THE CONSOL PROM AND PROVISION IS MADE FOR CASSETTE
C000                    0029 *  TAPE LOAD OF PROGRAMS OR DATA.
C000                    0030 *
C000                    0031 *
C000                    0032 *  NOTE: CONSOL,SOLOS AND SOLED ARE REGISTERED TRADEMARKS
C000                    0033 *        OF:
C000                    0034 *              PROCESSOR TECHNOLOGY CORP.
C000                    0035 *                 EMERYVILLE,CALIF
C000                    0036 *
C000                    0037 *    THE FOLLOWING CODE IS THE PROPERTY OF PROCESSOR
C000                    0038 *  TECHNOLOGY CORP.  IT IS DISTRIBUTED ON A 'PERSONAL USE'
C000                    0039 *  BASIS FOR THE BENIFIT OF SOL SYSTEM OWNERS.  ALL FORMS
C000                    0040 *  OF THE CODE ARE COPYRIGHT 1976 BY PROCESSOR TECHNOLOGY
C000                    0041 *  AND ALL RIGHTS THERIN ARE RESERVED.
C000                    0042 *
C000                    0043 *
C000                    0044 *
C000                    0045 *    AUTO-STARTUP CODE
C000                    0046 *
C000 00                 0047          DB        0           FOUR PHASE WONDER
C001 C3 65 C2           0048 BGIN     JMP       STRTA
C004                    0049 *
C004                    0050 *
C004                    0051 *
C004                    0052 *                    -----  SOL SYSTEM I/O ROUTINES  *-----
C004                    0053 *
C004                    0054 *    THE FOLLOWING CODE IS STANDARDIZED FOR ALL SOL SYSTEM
C004                    0055 *    SOLFTWARE IT PROVIDES COMMON ENTRY POINTS FOR INPUT AND
C004                    0056 *    OUTPUT OPERATIONS.  CONSOL DOES NOT HAVE PROVISION FOR
C004                    0057 *    PARALLEL I/O OPERATIONS BECAUSE OF SPACE LIMITATIONS.
C004                    0058 *
C004                    0059 *
C004                    0060 *
C004                    0061 *
C004                    0062 *              JUMP TABLE INPUT/OUTPUT ROUTINES
C004                    0063 *
C004                    0064 *    THIS ROUTINE OUTPUTS THE CHARACTER IN REGISTER 'B' TO
C004                    0065 *  THE OUTPUT DEVICE POINTED TO BY THE CURRENT OUTPUT SELECT
C004                    0066 *  REGISTER.  THE DEVICES ARE DEFINED AS FOLLOWS:
C004                    0067 *
C004                    0068 *        0 - DIS SCREEN
C004                    0069 *        1 - SERIAL OUTPUT PORT
C004                    0070 *        2 - PARALLEL OUTPUT PORT  (NOT AVAILABLE ON CONSOL)
C004                    0071 *        3 - ERROR HANDLER
C004                    0072 *
C004                    0073 *    ENTRY AT:  SOUT SELECTS CURRENT OUTPUT DEVICE
C004                    0074 *               AOUT SELECTS DEVICE IN REGISTER 'A'
C004                    0075 *
C004 3A 03 C8           0076 SOUT     LDA       OPORT       GET PORT NUMBER FROM MEMORY LOCATION
C007 E6 03              0077 AOUT     ANI       3           KEEP IT IN CONTROL
C009 E5                 0078          PUSH      H           WE'LL RESTORE IT LATER
C00A 21 84 C1           0079          LXI       H,OTAB      POINT TO TABLE
C00D 07                 0080          RLC       .           COMPUTE ADDRESS
C00E 85                 0081          ADD       L
C00F 6F                 0082          MOV       L,A         WE HAVE IT
C010 C3 71 C0           0083          JMP       DISPT       GO TO HL....
C013                    0084 *
C013                    0085 *
C013                    0086 *
C013                    0087 *    THIS ROUTINE INPUTS A CHARACTER TO REGISTER 'A' FROM
C013                    0088 *  THE CURRENT INPUT DEVICE POINTED TO BY THE CURRENT INPUT
C013                    0089 *  SELECT REGISTER.
C013                    0090 *
C013                    0091 *    ENTRY POINTS ARE DEFINED:
C013                    0092 *
C013                    0093 *          0 - KEYBOARD INPUT
C013                    0094 *          1 - SERIAL INPUT
C013                    0095 *          2 - PARALLEL INPUT (NOT AVAILABLE ON CONSOL)
C013                    0096 *          3 - ERROR HANDLER
C013                    0097 *
C013 3A 04 C8           0098 SINP     LDA       IPORT       GET PORT NUMBER FROM MEMORY LOCATION
C016 E6 03              0099 AINP     ANI       3           WE MUST BE REASONABLE
C018 E5                 0100          PUSH      H           SAVE H&L
C019 21 BC C1           0101          LXI       H,ITAB      POINT TO TABLE
C01C 07                 0102          RLC       .           THE MATH
C01D 85                 0103          ADD       L
C01E 6F                 0104          MOV       L,A         DONE
C01F C3 71 C0           0105          JMP       DISPT       WE HAVE THE ADDRESS...GO TO HEAVEN
C022                    0106 *
C022                    0107 *
C022                    0108 *       KEYBOARD INPUT STATUS CHECK
C022                    0109 *
C022                    0110 *  THIS ROUTINE TESTS THE KEYBOARD STATUS AND RETURNS
C022                    0111 *  WITH THE TEST BITS SET.
C022                    0112 *
C022 DB FA              0113 KSTAT    IN        STAPT       GET STATUS WORD
C024 E6 01              0114          ANI       KDR         TEST KEYBOARD BIT
C026 C9                 0115          RET                   FLAGS ARE SET
C027                    0116 *
C027                    0117 *
C027                    0118 *       KEYBOARD DATA INPUT
C027                    0119 *
C027                    0120 *  THIS ROUTINE, ENTRY AT KREAD, GETS THE DATA FROM THE
C027                    0121 *  KEYBOARD.  ON RETURN THE CHARACTER IS IN REGISTER 'A'.
C027                    0122 *
C027 E1                 0123 KREA1    POP       H           JUMP TABLE ENTRY POINT
C028 CD 22 C0           0124 KREAD    CALL      KSTAT       CHECK STATUS
C02B C2 28 C0           0125          JNZ       KREAD       WAIT FOR INPUT
C02E DB FC              0126          IN        KDATA       GET DATA
C030 C9                 0127          RET                   GO BACK WITH IT
C031                    0128 *
C031                    0129 *
C031                    0130 *    SERIAL INPUT STATUS CHECK
C031                    0131 *
C031 DB F8              0132 SSTAT    IN        SERST       GET SERIAL STATUS WORD
C033 E6 40              0133          ANI       SDR         TEST FOR SERIAL DATA READY
C035 C9                 0134          RET                   FLAGS ARE SET
C036                    0135 *
C036                    0136 *
C036                    0137 *    SERIAL DATA INPUT
C036                    0138 *
C036 E1                 0139 SKEA1    POP       H           RESTORE HL FROM JUMP TABLE ENTRY
C037 CD 31 C0           0140 SKEAD    CALL      SSTAT       NORMAL ENTRY POINT
C03A CA 37 C0           0141          JZ        SKEAD       WAIT FOR INPUT
C03D DB F9              0142          IN        SDATA       GET DATA BYTE
C03F C9                 0143          RET                   WE HAVE IT
C040                    0144 *
C040                    0145 *
C040                    0146 *    SERIAL DATA OUTPUT
C040                    0147 *
C040 E1                 0148 SEROT    POP       H           JUMP TABLE ENTRY POINT
C041 DB F8              0149 SDROT    IN        SERST       GET PORT STATUS
C043 17                 0150          RAL       .           PUT HIGH BIT IN CARRY
C044 D2 41 C0           0151          JNC       SDROT       LOOP UNTIL TRANSMITTER BUFFER IS EMPTY
C047 78                 0152          MOV       A,B         GET THE CHARACTER BACK
C048 D3 F9              0153          OUT       SDATA       SEND IT OUT
C04A C9                 0154          RET                   AND WE'RE DONE
C04B                    0155 *
C04B                    0156 *
C04B                    0157 *
C04B                    0158 *
C04B                    0159 *
```

```
C048                    0160 *              VIDEO DISPLAY DRIVER ROUTINES
C048                    0161 *
C048                    0162 *
C048                    0163 *  THESE ROUTINES ALLOW FOR STANDARD VIDEO TERMINAL
C048                    0164 *  OPERATIONS.  ON ENTRY, THE CHARACTER FOR OUTPUT IS IN
C048                    0165 *  REGISTER B AND ALL REGISTERS ARE UNALTERED ON RETURN.
C048                    0166 *
C048                    0167 *     THE 'CONSOL' VERSION OF THIS ROUTINE IS A MINIMUM
C048                    0168 *  IMPLEMENTATION OF ROUTINES ORIGINATED BY:
C048                    0169 *
C048                    0170 *                  IAN KETTLEBOROUGH
C048                    0171 *                        OF
C048                    0172 *              COLLEGE STATION, TEXAS
C048                    0173 *
C048                    0174 *    SOLOS AND SOLED CONTAIN THE ESC SEQUENCES AND OTHER
C048                    0175 *  FULL IMPLEMENTATION FEATURES.
C048                    0176 *
C048 E5                 0177 VDMOT    PUSH    H           SAVE EVERYBODY
C04C D5                 0178 VDMO1    PUSH    D           ENTRY FROM DEVICE SELECT
C04D C5                 0179          PUSH    B
C04E F5                 0180          PUSH    PSW
C04F 78                 0181          MOV     A,B         SAVE IN B...STRIP PARITY BEFORE SCREEN!
C050 21 65 C1           0182          LXI     H,TBL
C053 CD 62 C0           0183          CALL    TSRCH       GO PROCESS
C056                    0184 *
C056 CD 21 C1           0185 GOBACK   CALL    VDADD       GET SCREEN ADDRESS
C059 7E                 0186          MOV     A,M
C05A F6 80              0187          ORI     80H
C05C 77                 0188          MOV     M,A         CURSOR IS BACK ON
C05D F1                 0189 GOBK     POP     PSW
C05E C1                 0190          POP     B
C05F D1                 0191          POP     D           RESTORE ALL REGISTERS
C060 E1                 0192          POP     H
C061 C9                 0193          RET     .           EXIT FROM VDMOT
C062                    0194 *
C062                    0195 *
C062 7E                 0196 TSRCH    MOV     A,M         GET CHR FROM TABLE
C063 B7                 0197          ORA     A
C064 CA 7A C0           0198          JZ      CHAR        ZERO IS THE LAST
C067 B8                 0199          CMP     B           TEST THE CHR
C068 23                 0200          INX     H           POINT FORWARD
C069 C2 76 C0           0201          JNZ     NEXT
C06C E5                 0202          PUSH    H           FOUND ONE...SAVE ADDRESS
C06D C0 3D C1           0203          CALL    CREM        REMOVE CURSOR
C070 E1                 0204          POP     H
C071                    0205 *
C071                    0206 *
C071                    0207 *    THIS ROUTINE DISPATCHES TO THE ADDRESS POINTED TO
C071                    0208 *  BY THE HL REGISTER PAIR.  THE RETURN ADDRESS IS THE
C071                    0209 *  LAST ENTRY ON THE STACK.
C071                    0210 *
C071 7E                 0211 DISPT    MOV     A,M         GET LOW BYTE
C072 23                 0212          INX     H
C073 66                 0213          MOV     H,M         AND THE HIGH
C074 6F                 0214          MOV     L,A         WE HAVE PLACED THEM BOTH
C075 E9                 0215          PCHL                GO TO IT
C076                    0216 *
C076                    0217 *
C076 23                 0218 NEXT     INX     H           GO TO NEXT
C077 23                 0219          INX     H
C078 C3 62 C0           0220          JMP     TSRCH
C07B                    0221 *
C07B                    0222 *
C07B 78                 0223 CHAR     MOV     A,B         GET CHARACTER
C07C B7                 0224          ORA     A
C07D C8                 0225          RZ      .           RETURN IF A NULL
C07E FE 7F              0226          CPI     7FH         IS IT A DEL?
C080 C8                 0227          RZ      .           GO BACK IF SO
C081                    0228 *
C081                    0229 *
C081                    0230 *
C081 CD 21 C1           0231 OCHAR    CALL    VDADD       GET SCREEN ADDRESS
C084 78                 0232          MOV     A,B         GET CHARACTER FOR OUTPUT
C085 E6 7F              0233          ANI     7FH         NO HIGH BITS GO PAST HERE
C087 77                 0234          MOV     M,A         PUT CHR ON SCREEN
C088 3A 80 C8           0235          LDA     NCHAR       GET CHARACTER POSITION
C08B FE 3F              0236          CPI     63          END OF LINE?
C08D DA AD C0           0237          JC      OK
C090 3A 81 C8           0238          LDA     LINE
C093 FE 0F              0239          CPI     15          END OF SCREEN?
C095 C2 AD C0           0240          JNZ     OK
C098                    0241 *
C098                    0242 *   END OF SCREEN...ROLL UP ONE LINE
C098                    0243 *
C098 AF                 0244 SCROLL   XRA     A
C099 32 80 C8           0245          STA     NCHAR       BACK TO FIRST CHAR POSITION
C09C 4F                 0246 SROL     MOV     C,A
C09D CD 28 C1           0247          CALL    VDAD        CALCULATE LINE TO BE BLANKED
C0A0 AF                 0248          XRA     A
C0A1 CD 69 C0           0249          CALL    CLIN1       CLEAR IT
C0A4 3A 82 C8           0250          LDA     BOT
C0A7 3C                 0251          INR     A
C0A8 E6 0F              0252          ANI     0FH
C0AA C3 DD C0           0253          JMP     ERASJ
C0AD                    0254 *
C0AD                    0255 *   INCREMENT LINE COUNTER IF NECESSARY
C0AD                    0256 *
C0AD 3A 80 C8           0257 OK       LDA     NCHAR       GET CHR POSITION
C0B0 3C                 0258          INR     A
C0B1 32 80 C8           0259          STA     NCHAR       STORE THE NEW
C0B4 FE 40              0260          CPI     64
C0B6 D8                 0261          RC                  WE'RE PAST THE END... REWIND THE COUNT
C0B7 AF                 0262          XRA     A
C0B8 32 80 C8           0263          STA     NCHAR
C0BB 3A 81 C8           0264          LDA     LINE        GET THE LINE COUNT
C0BE 3C                 0265          INR     A
C0BF E6 0F              0266          ANI     0FH         MOD 15 INCREMENT
C0C1 32 81 C8           0267 CUR      STA     LINE        STORE THE NEW
C0C4 C9                 0268          RET
C0C5                    0269 *
C0C5                    0270 *    ERASE SCREEN
C0C5                    0271 *
C0C5 21 00 CC           0272 PERSE    LXI     H,VDMEM     POINT TO SCREEN
C0C8 36 A0              0273          MVI     M,80H+' '   THIS IS THE CURSOR
C0CA                    0274 *
C0CA 23                 0275 ERAS1    INX     H           BUMP THE COUNT
C0CB 7C                 0276          MOV     A,H         GET HIGH ORDER
C0CC FE D0              0277          CPI     0D0H        THE TOP
C0CE D2 D6 C0           0278          JNC     ERAS2
C0D1 36 20              0279          MVI     M,' '       PUT IN A BLANK
C0D3 C3 CA C0           0280          JMP     ERAS1
C0D6                    0281 *
C0D6 AF                 0282 ERAS2    XRA     A
C0D7 32 81 C8           0283          STA     LINE        ZERO LINE
C0DA 32 80 C8           0284          STA     NCHAR       LEFT SIDE OF SCREEN
C0DD                    0285 *
C0DD D3 FE              0286 ERAS3    OUT     0STAT       RESET SCROLL PARAMETERS
C0DF 32 82 C8           0287          STA     BOT         BEGINNING OF TEXT OFFSET
C0E2 C9                 0288          RET
C0E3                    0289 *
C0E3                    0290 *
C0E3 CD 21 C1           0291 CLINE    CALL    VDADD       GET CURRENT SCREEN ADDRESS
C0E6 3A 80 C8           0292          LDA     NCHAR       CURRENT CURSOR POSITION
C0E9 FE 40              0293 CLIN1    CPI     64          NO MORE THAN 63
C0EB D8                 0294          RNC     .           ALL DONE
C0EC 36 20              0295          MVI     M,' '       ALL SPACED OUT
C0EE 23                 0296          INX     H
C0EF 3C                 0297          INR     A
C0F0 C3 E9 C0           0298          JMP     CLIN1       LOOP TO END OF LINE
C0F3                    0299 *
C0F3                    0300 *    HOME CURSOR
C0F3                    0301 *
C0F3 AF                 0302 PHOME    XRA     A
C0F4 32 80 C8           0303          STA     NCHAR
C0F7 C3 C1 C0           0304          JMP     CUR
C0FA                    0305 *
C0FA                    0306 *  MOVE CURSOR DOWN ONE LINE
C0FA                    0307 *
C0FA 3A 81 C8           0308 PDOWN    LDA     LINE
C0FD FE 0F              0309          CPI     15
C0FF C8                 0310          RZ      .           HOW FAR IS DOWN?
C100 3C                 0311          INR     A
C101 C3 C1 C0           0312          JMP     CUR
C104                    0313 *
C104                    0314 *  ROUTINE TO MOVE THE CURSOR UP ONE LINE
C104                    0315 *
C104 3A 81 C8           0316 PUP      LDA     LINE        GET LINE COUNT
C107 B7                 0317          ORA     A
C108 C8                 0318          RZ      .           DON'T GO MORE UP THAN UP
C109 3D                 0319          DCR     A
C10A C3 C1 C0           0320          JMP     CUR
C10D                    0321 *
C10D                    0322 *  MOVE CURSOR LEFT ONE POSITION
C10D                    0323 *
C10D 3A 80 C8           0324 PLEFT    LDA     NCHAR
C110 B7                 0325          ORA     A
C111 C8                 0326          RZ      .           DON'T GO MORE BACK THAN BACK
C112 3D                 0327          DCR     A
C113 32 80 C8           0328 PCUR     STA     NCHAR
C116 C9                 0329          RET
C117                    0330 *
C117                    0331 *     CURSOR RIGHT ONE POSITION
C117                    0332 *
C117 3A 80 C8           0333 PRIT     LDA     NCHAR
C11A FE 3F              0334          CPI     63
C11C C8                 0335          RZ                  HOW RIGHT CAN WE BE?
C11D 3C                 0336          INR     A
C11E C3 13 C1           0337          JMP     PCUR
C121                    0338 *
C121                    0339 *   ROUTINE TO CALCULATE SCREEN ADDRESS
C121                    0340 *
C121                    0341 *   ENTRY AT:    RETURNS:
C121                    0342 *
C121                    0343 *         VDADD  CURRENT SCREEN ADDRESS
C121                    0344 *         VDAD2  ADDRESS OF CURRENT LINE, CHAR 'C'
C121                    0345 *         VDAD   LINE 'A', CHARACTER POSITION 'C'
C121                    0346 *
C121 3A 80 C8           0347 VDADD    LDA     NCHAR       GET CHARACTER POSITION
C124 4F                 0348          MOV     C,A         'C' KEEPS IT
C125 3A 81 C8           0349 VDAD2    LDA     LINE        LINE POSITION
C128 6F                 0350 VDAD     MOV     L,A         INTO 'L'
C129 3A 82 C8           0351          LDA     BOT         GET TEXT OFFSET
C12C 85                 0352          ADD     L           ADD IT TO THE LINE POSITION
C12D 6F                 0353          RHC     .           TIMES TWO
C12E 6F                 0354          RHC     .           MAKES FOUR
C12F 6F                 0355          MOV     L,A         L HAS IT
C130 E6 03              0356          ANI     3           MOD THREE FOR LATER
C132 57                 0357          MOV     D,A
C133 3E CC              0358          MVI     A,<VDMEM    LOW SCREEN OFFSET
C135 82                 0359          ADD     D
C136 67                 0360          MOV     H,A         NOW H IS DONE
C137 7D                 0361          MOV     A,L         TWIST L'S ARM
C138 E6 C0              0362          ANI     0C0H
C13A 81                 0363          ADD     C
C13B 6F                 0364          MOV     L,A
C13C C9                 0365          RET                 H & L ARE NOW PERVERTED
C13D                    0366 *
C13D                    0367 *    ROUTINE TO REMOVE CURSOR
C13D                    0368 *
C13D CD 21 C1           0369 CREM     CALL    VDADD       GET CURRENT SCREEN ADDRESS
C140 7E                 0370          MOV     A,M
C141 E6 7F              0371          ANI     7FH         STRIP OFF THE CURSOR
C143 77                 0372          MOV     M,A
C144 C9                 0373          RET
C145                    0374 *
C145                    0375 *     ROUTINE TO BACKSPACE
C145                    0376 *
C145 CD 0D C1           0377 VBACK    CALL    PLEFT
C148 CD 21 C1           0378          CALL    VDADD       GET SCREEN ADDRESS
C14B 36 20              0379          MVI     M,' '       PUT A BLANK THERE
C14D C9                 0380          RET
C14E                    0381 *
C14E                    0382 *     ROUTINE TO PROCESS A CARRIAGE RETURN
C14E                    0383 *
C14E CD E3 C0           0384 PCR      CALL    CLINE       CLEAR FROM CURRENT CURSOR TO END OF LINE
C151 AF                 0385          XRA     A           REWIND IT
C152 C3 13 C1           0386          JMP     PCUR        AND STORE THE NEW VALUE
C155                    0387 *
C155                    0388 *   ROUTINE TO PROCESS LINEFEED
C155                    0389 *
C155 3A 81 C8           0390 PLF      LDA     LINE        GET LINE COUNT
C158 FE 0F              0391          CPI     15          ARE WE AT THE BOTTOM?
C15A D2 61 C1           0392          JNC     SC
C15D 3C                 0393          INR     A
C15E C3 C1 C0           0394          JMP     CUR         ONE MORE LINE UP
C161                    0395 *
C161 AF                 0396 SC       XRA     A
C162 C3 9C C0           0397          JMP     SROL
C165                    0398 *
C165                    0399 *
C165                    0400 *
C165                    0401 *     THIS TABLE DEFINES THE CHARACTERS FOR SPECIAL
C165                    0402 * PROCESSING.  IF THE CHARACTER IS NOT IN THE TABLE IT
C165                    0403 * GOES TO THE SCREEN.
C165                    0404 *
C165 0C                 0405 TBL      DB      CLEAR       SCREEN
C166 C5 C0              0406          DW      PERSE
C168 97                 0407          DB      UP          CURSOR
C169 D4 C1              0408          DW      PUP
C16B 9A                 0409          DB      DOWN        "
C16C FA C0              0410          DW      PDOWN
C16E 81                 0411          DB      LEFT
C16F 0D C1              0412          DW      PLEFT
C171 93                 0413          DB      RIGHT       "
C172 17 C1              0414          DW      PRIT
C174 8E                 0415          DB      HOME        "
C175 F3 C0              0416          DW      PHOME
C177 0D                 0417          DB      CR          CARRIAGE RETURN
C178 4E C1              0418          DW      PCR
C17A 0A                 0419          DB      LF          LINE FEED
C17B 55 C1              0420          DW      PLF
C17D 5F                 0421          DB      BACKS       BACK SPACE
C17E 45 C1              0422          DW      PBACK
C180 8F                 0423          DB      MODE        MODE KEY
C181 B4 C1              0424          DW      COMND
C183 00                 0425          DB      0           END OF TABLE
C184                    0426 *
C184                    0427 *   OUTPUT DEVICE TABLE
C184                    0428 *
C184 4C C0              0429 OTAB     DW      VDMO1       DISPLAY DRIVER
C186 4C C0              0430          DW      SEROT       SERIAL OUTPUT
C188 AD C1              0431          DW      ERROT       ERROR HANDLER (FOR CONSOL)
C18A AD C1              0432          DW      ERROT       ERROR HANDLER
C18C                    0433 *
C18C                    0434 *   INPUT DEVICE TABLE
C18C                    0435 *
C18C 27 C0              0436 ITAB     DW      KREAL       KEYBOARD INPUT
C18E 36 C0              0437          DW      SREAL       SERIAL INPUT
C190 AD C1              0438          DW      ERROT       ERROR HANDLER (FOR CONSOL)
C192 AD C1              0439          DW      ERROT       ERROR HANDLER
C194                    0440 *
C194                    0441 *
C194                    0442 *              COMMAND TABLE
C194                    0443 *
C194                    0444 *  THIS TABLE DESCRIBES THE VALID COMMANDS FOR CONSOL
C194                    0445 *
C194 54 45              0446 COMTAB   ASC     'TE'
C196 70 C2              0447          DW      TERM
C198 44 55              0448          ASC     'DU'
C19A A1 C2              0449          DW      DUMP
C19C 45 4E              0450          ASC     'EN'
C19E 07 C3              0451          DW      ENTER
C1A0 45 58              0452          ASC     'EX'
C1A2 35 C3              0453          DW      EXEC
C1A4 54 4C              0454          ASC     'TL'
C1A6 39 C3              0455          DW      TLOAD
C1A8 42 41              0456          ASC     'BA'        SPECIAL COMMAND TO EXECUTE @
C1AA 00 00              0457          DW      @
C1AC 00                 0458          DB      @           END OF TABLE MARK
C1AD                    0459 *
C1AD                    0460 *
C1AD                    0461 *              CONSOL PORT ERROR HANDLER
C1AD                    0462 *
C1AD AF                 0463 ERROT    XRA     A
C1AE 32 64 C8           0464          STA     IPORT       DEFAULT TO SCREEN
C1B1 32 03 C8           0465          STA     OPORT       DEFAULT TO SCREEN
C1B4                    0466 *
C1B4                    0467 *
C1B4                    0468 *
C1B4                    0469 *
C1B4                    0470 *              ---  COMMAND MODE  ---
C1B4                    0471 *
C1B4                    0472 *
C1B4                    0473 *    THIS ROUTINE GETS AND PROCESSES COMMANDS
C1B4                    0474 *
C1B4 31 00 CC           0475 COMND    LXI     SP,SYSTP    SET STACK POINTER
C1B7 CD 12 C2           0476          CALL    PROMPT      PUT PROMPT ON SCREEN
C1BA CD C3 C1           0477          CALL    GCLIN       GET COMMAND LINE
C1BD CD E1 C1           0478          CALL    COPRC       PROCESS THE LINE
C1C0 C3 B4 C1           0479          JMP     COMND       OVER AND OVER
C1C3                    0480 *
C1C3                    0481 *
C1C3                    0482 *
C1C3                    0483 *   THIS ROUTINE READS A COMMAND LINE FROM THE SYSTEM
C1C3                    0484 *  KEYBOARD AND PROCESSES IT TO THE SCREEN.
C1C3                    0485 *
C1C3                    0486 *  C/R   TERMINATES THE SEQUENCE ERASING ALL CHARS TO THE
C1C3                    0487 *        RIGHT OF THE CURSOR
C1C3                    0488 *  L/F   TERMINATES THE SEQUENCE
C1C3                    0489 *  MODE  RESTARTS THE COMMAND LINE.
C1C3                    0490 *
C1C3 CD 28 C0           0491 GCLIN    CALL    KREAD       READ KEYBOARD
C1C6 FE 20              0492          CPI     20H
C1C8 47                 0493          MOV     B,A
C1C9 DA D2 C1           0494          JC      PROCS       PROCESS CONTROL CHARACTER
C1CC CD 4B C0           0495 CONT     CALL    VDMOT
C1CF C3 C3 C1           0496          JMP     GCLIN
C1D2                    0497 *
C1D2                    0498 *    PROCESS CONTROL KEYS
C1D2                    0499 *
C1D2 FE 00              0500 PROCS    CPI     CR
C1D4 CA D9 C1           0501          JZ      CRPRC       ERASE THE REMAINING LINE PRIOR TO RETURN
C1D7 FE 0A              0502          CPI     LF
C1D9 C8                 0503          RZ      .           IF SO GO PROCESS
C1DA C3 C3 C1           0504          JMP     GCLIN       NO CONTROL CHARS TO SCREEN
C1DD                    0505 *
C1DD                    0506 *
C1DD CD E3 C0           0507 CRPRC    CALL    CLINE       CLEAR REMAINING LINE
C1E0 C9                 0508          RET                 NOW PROCESS
C1E1                    0509 *
C1E1                    0510 *
C1E1                    0511 *       FIND AND PROCESS COMMAND
```

```
C1E1                    0512 *
C1E1 CD 3D C1           0513 COPRC      CALL    CREM            REMOVE THE CURSOR
C1E4 0E 01              0514            MVI     C,1             SET FOR CHARACTER POSITION
C1E6 CD 25 C1           0515            CALL    VDAD2           GET SCREEN ADDRESS
C1E9 EB                 0516            XCHG
C1EA CD 32 C2           0517            CALL    SCHR            SCAN PAST BLANKS
C1ED CA 8E C3           0518            JZ      ERR1            NO COMMAND?
C1F0 EB                 0519            XCHG    .               HL HAS FIRST CHR
C1F1 11 94 C1           0520            LXI     D,COMTAB        POINT TO COMMAND TABLE
C1F4                    0521 *
C1F4                    0522 *  THIS ROUTINE SEARCHES THROUGH A TABLE, POINTED TO
C1F4                    0523 *  BY 'DE', FOR A DOUBLE CHARACTER MATCH OF THE 'HL'
C1F4                    0524 *  MEMORY CONTENT.  IF NO MATCH IS FOUND THE SCAN ENDS
C1F4                    0525 *  BY PLACING A QUESTION MARK WITHIN THE SEARCH STRING.
C1F4                    0526 *
C1F4 1A                 0527 FDCOM      LDAX    D
C1F5 B7                 0528            ORA     A               TEST FOR TABLE END
C1F6 CA 8F C3           0529            JZ      ERR2            NOT FOUND..COMMAND ERROR
C1F9 E5                 0530            PUSH    H               SAVE START OF SCAN ADDRESS
C1FA BE                 0531            CMP     M               TEST FIRST CHR
C1FB 13                 0532            INX     D
C1FC C2 08 C2           0533            JNZ     NCOM
C1FF                    0534 *
C1FF 23                 0535            INX     H
C200 1A                 0536            LDAX    D
C201 BE                 0537            CMP     M               NOW SECOND CHARACTER
C202 C2 08 C2           0538            JNZ     NCOM            GOODNESS
C205                    0539 *
C205 C1                 0540            POP     B               CLEAR THE STACK
C206 EB                 0541            XCHG    .               DE HAS SCAN ADDRESS
C207 23                 0542            INX     H               HL HAS COMMAND ADDRESS
C208 C3 71 C0           0543            JMP     DISPT           DISPATCH TO IT
C20B                    0544 *
C20B                    0545 *
C20B 13                 0546 NCOM       INX     D               GO TO NEXT ENTRY
C20C 13                 0547            INX     D
C20D 13                 0548            INX     D
C20E E1                 0549            POP     H               GET BACK ORIGINAL ADDRESS
C20F C3 F4 C1           0550            JMP     FDCOM           CONTINUE SEARCH
C212                    0551 *
C212                    0552 *
C212                    0553 *      OUTPUT A CRLF FOLLOWED BY A PROMPT
C212                    0554 *  ( WITH CONSOL ALL OPERATIONS ARE ON THE SCREEN)
C212                    0555 *
C212 CD 1A C2           0556 PROMPT     CALL    CRLF
C215 06 3L              0557            MVI     B,'>'           THE PROMPT
C217 C3 4B C0           0558            JMP     VDMOT           PUT IT ON THE SCREEN
C21A                    0559 *
C21A 06 0A              0560 CRLF       MVI     B,LF            LINE FEED
C21C CD 4B C0           0561            CALL    VDMOT
C21F 06 0D              0562            MVI     B,CR            CARRIAGE RETURN
C221 C3 4B C0           0563            JMP     VDMOT           PUT IT OUT AND RETURN
C224                    0564 *
C224                    0565 *
C224                    0566 *  SCAN OVER UP TO 12 CHARACTERS LOOKING FOR A BLANK
C224                    0567 *
C224 0E 0C              0568 SBLK       MVI     C,12            MAXIMUM COMMAND STRING
C226 1A                 0569 SBLK1      LDAX    D
C227 FE 2d              0570            CPI     BLANK
C229 CA 32 C2           0571            JZ      SCHR            GOT A BLANK NOW SCAN PAST IT
C22C 13                 0572            INX     D
C22D 0D                 0573            DCR     C               NO MORE THAN TWELVE
C22E C2 26 C2           0574            JNZ     SBLK1
C231 C9                 0575            RET                     GO BACK WITH ZERO FLAG SET
C232                    0576 *
C232                    0577 *
C232                    0578 *  SCAN PAST UP TO 10 BLANK POSITIONS LOOKING FOR
C232                    0579 * A NON BLANK CHARACTER.
C232                    0580 *
C232 0E 0A              0581 SCHR       MVI     C,10            SCAN TO FIRST NON BLANK CHR WITHIN 10
C234 1A                 0582 SCHR1      LDAX    D               GET NEXT CHARACTER
C235 FE 20              0583            CPI     SPACE
C237 C0                 0584            RNZ     .               WE'RE PAST THEM
C238 13                 0585            INX     D               NEXT SCAN ADDRESS
C239 0D                 0586            DCR     C
C23A C8                 0587            RZ      .               COMMAND ERROR
C23B C3 34 C2           0588            JMP     SCHR1           KEEP LOOPING
C23E                    0589 *
C23E                    0590 *  THIS ROUTINE SCANS OVER CHARACTERS, PAST BLANKS AND
C23E                    0591 * CONVERTS THE FOLLOWING ADDRESS TO HEX.  ERRORS RETURN TO
C23E                    0592 * THE ERROR HANDLER.
C23E                    0593 *
C23E CD 24 C2           0594 SCONV      CALL    SBLK
C241 CA 8E C3           0595            JZ      ERR1
C244                    0596 *
C244                    0597 *  THIS ROUTINE CONVERTS ASCII DIGITS INTO BINARY FOLLOWING
C244                    0598 * A STANDARD HEX CONVERSION.  THE SCAN STOPS WHEN AN ASCII
C244                    0599 * SPACE IS ENCOUNTERED.  PARAMETER ERRORS REPLACE THE ERROR
C244                    0600 * CHARACTER ON THE SCREEN WITH A QUESTION MARK.
C244                    0601 *
C244 21 00 00           0602 SHEX       LXI     H,0             CLEAR H & L
C247 1A                 0603 SHE1       LDAX    D               GET CHARACTER
C248 FE 20              0604            CPI     20H             IS IT A SPACE?
C24A C8                 0605            RZ      .               IF SO
C24B                    0606 *
C24B 29                 0607 HCONV      DAD     H               MAKE ROOM FOR THE NEW ONE
C24C 29                 0608            DAD     H
C24D 29                 0609            DAD     H
C24E 29                 0610            DAD     H
C24F CD 5B C2           0611            CALL    HCOV1           DO THE CONVERSION
C252 D2 8E C3           0612            JNC     ERR1            NOT VALID HEXIDECIMAL VALUE
C255 85                 0613            ADD     L
C256 6F                 0614            MOV     L,A             MOVE IT IN
C257 13                 0615            INX     D               BUMP THE POINTER
C258 C3 47 C2           0616            JMP     SHE1
C25B                    0617 *
C25B D6 30              0618 HCOV1      SUI     48              REMOVE ASCII BIAS
C25D FE 0A              0619            CPI     10
C25F D8                 0620            RC      .               IF LESS THAN 9
C260 D6 07              0621            SUI     7               IT'S A LETTER??
C262 FE 10              0622            CPI     10H
C264 C9                 0623            RET                     WITH TEST IN HAND
C265                    0624 *
C265                    0625 *
C265                    0626 *    SYSTEM START UP, CLEAR PART OF RAM AND SET STACK
C265                    0627 * POINTER, FALLING THROUGH TO TERMINAL MODE.
C265                    0628 *
C265 AF                 0629 STRTA      XRA     A
C266 4F                 0630            MOV     C,A             WE CLEAR THE FIRST 256 BYTES
C267 21 00 C8           0631            LXI     H,SYSRAM        POINT TO SYSTEM RAM
C26A                    0632 *
C26A 77                 0633 CLERA      MOV     M,A
C26B 23                 0634            INX     H
C26C 0C                 0635            INR     C
C26D C2 6A C2           0636            JNZ     CLERA           CLEAR FIRST 256 BYTES
C270                    0637 *
C270                    0638 *
C270                    0639 *
C270                    0640 *                  TERM COMMAND
C270                    0641 *
C270                    0642 *  THIS ROUTINE GETS CHARACTERS FROM THE SYSTEM KEYBOARD
C270                    0643 *  AND OUTPUTS THEM TO THE SERIAL OUTPUT PORT.  IT IS
C270                    0644 *  INTENDED TO CONFIGURE THE SOL AS A STANDARD VIDEO
C270                    0645 *  TERMINAL.  COMMAND KEYS ARE NOT OUTPUT TO THE OUTPUT
C270                    0646 *  PORT BUT ARE INTERPRETED AS DIRECT SOL COMMANDS.
C270                    0647 *  THE MODE COMMAND, RECEIVED BY THE KEYBOARD, PUTS THE
C270                    0648 *  SOL IN THE COMMAND MODE.
C270                    0649 *
C270                    0650 *
C270 31 00 CC           0651 TERM       LXI     SP,SYSTP        SET STACK POINTER
C273 CD ED C3           0652            CALL    TOFF            SLOW DOWN THE TAPES
C276 CD C5 C0           0653            CALL    PERSE           CLEAR THE SCREEN
C279                    0654 *
C279 CD 22 C0           0655 KIN        CALL    KSTAT           IS THERE ONE WAITING?
C27C C2 94 C2           0656            JNZ     TIN
C27F 00 FC              0657            IN      KDATA           GET THE CHARACTER
C281 47                 0658            MOV     B,A
C282 E6 80              0659            ANI     80H             COMMAND KEY?
C284 CA 8D C2           0660            JZ      TOUT
C287 CD 4B C0           0661            CALL    VDMOT           PROCESS IT
C28A C3 90 C2           0662            JMP     TIN
C28D                    0663 *
C28D CD 41 C0           0664 TOUT       CALL    SDROT           OUTPUT IT TO THE SERIAL PORT
C290 CD 31 C0           0665 TIN        CALL    SSTAT           GET SERIAL STATUS
C293 CA 79 C2           0666            JZ      KIN             LOOP IF NOT
C296 DB F9              0667            IN      SDATA           GET DATA
C298 E6 7F              0668            ANI     7FH             NO HIGH BITS FROM HERE
C29A 47                 0669            MOV     B,A             IT'S OUTPUT FROM 'B'
C29B CD 4B C0           0670            CALL    VDMOT           PUT IT ON THE SCREEN
C29E C3 79 C2           0671            JMP     KIN             LOOP OVER AND OVER
C2A1                    0672 *
C2A1                    0673 *
C2A1                    0674 *
C2A1                    0675 *                  DUMP COMMAND
C2A1                    0676 *
C2A1                    0677 *    THIS ROUTINE DUMPS CHARACTERS FROM MEMORY TO THE
C2A1                    0678 *  CURRENT OUTPUT DEVICE.  (WITH CONSOL ALL OUTPUT GOES TO
C2A1                    0679 *  THE SCREEN).  ALL VALUES ARE DESPLAYED AS ASCII HEX.
C2A1                    0680 *
C2A1                    0681 *  THE COMMAND FORM IS AS FOLLOWS:
C2A1                    0682 *
C2A1                    0683 *          DUMP  ADDR1  ADDR2
C2A1                    0684 *
C2A1                    0685 *    THE VALUES FROM ADDR1 TO ADDR2 ARE THEN OUTPUT TO THE
C2A1                    0686 *  OUTPUT DEVICE.  IF ONLY ADDR1 IS SPECIFIED THEN THE
C2A1                    0687 *  VALUE AT THAT ADDRESS IS OUTPUT.
C2A1                    0688 *
C2A1 CD 3E C2           0689 DUMP       CALL    SCONV           SCAN TO FIRST ADDRESS AND CONVERT IT
C2A4 E5                 0690            PUSH    H               SAVE THE VALUE
C2A5 CD 32 C2           0691            CALL    SCHR            GET THE NEXT
C2A8 E1                 0692            POP     H
C2A9 CA B4 C2           0693            JZ      POVER           NO SECOND VALUE
C2AC E5                 0694            PUSH    H
C2AD CD 44 C2           0695            CALL    SHEX            GET SECOND
C2B0 D1                 0696            POP     D               THIS IS THE FIRST
C2B1 C3 B6 C2           0697            JMP     NPASS           MIND BENDERS
C2B4                    0698 *
C2B4 54                 0699 POVER      MOV     D,H             NO SECOND PARAMETER COPY FIRST TO DE
C2B5 5D                 0700            MOV     E,L
C2B6 EB                 0701 NPASS      XCHG    .               HL HAS START, DE HAS END
C2B7                    0702 *
C2B7 CD 1A C2           0703 DLOOP      CALL    CRLF
C2BA DB FC              0704            IN      KDATA
C2BC FE 80              0705            CPI     MODE            'MODE KEY' WILL ESCAPE THE DUMP
C2BE CA B4 C1           0706            JZ      COMND
C2C1 CD DD C2           0707            CALL    ADOUT           OUTPUT ADDRESS
C2C4 CD E5 C2           0708            CALL    BOUT            ANOTHER SPACE TO KEEP IT PRETTY
C2C7 0E 10              0709            MVI     C,16            VALUES PER LINE
C2C9                    0710 *
C2C9 7E                 0711 DLP1       MOV     A,M             GET THE CHR
C2CA C5                 0712            PUSH    B               SAVE VALUE COUNT
C2CB CD E2 C2           0713            CALL    HBOUT           SEND IT OUT WITH A BLANK
C2CE CD 02 C3           0714            CALL    ACOMP           COMPARE ADDRESSES
C2D1 D2 B4 C1           0715            JNC     COMND           ALL DONE
C2D4 C1                 0716            POP     B               VALUES PER LINE
C2D5 23                 0717            INX     H
C2D6 0D                 0718            DCR     C               BUMP THE LINE COUNT
C2D7 C2 C9 C2           0719            JNZ     DLP1            NOT ZERO IF MORE FOR THIS LINE
C2DA C3 B7 C2           0720            JMP     DLOOP           DO A LFCR BEFORE THE NEXT
C2DD                    0721 *
C2DD                    0722 *    OUTPUT HL AS HEX 16 BIT VALUE
C2DD                    0723 *
C2DD 7C                 0724 ADOUT      MOV     A,H             H FIRST
C2DE CD EA C2           0725            CALL    HEOUT
C2E1 7D                 0726            MOV     A,L             THEN L FOLLOWED BY A SPACE
C2E2                    0727 *
C2E2 CD EA C2           0728 HBOUT      CALL    HEOUT
C2E5 06 20              0729 BOUT       MVI     B,' '
C2E7 C3 4B C0           0730            JMP     VDMOT           CONSOL PUTS IT ON THE SCREEN
C2EA                    0731 *
C2EA 4F                 0732 HEOUT      MOV     C,A             GET THE CHARACTER
C2EB 0F                 0733            RRC
C2EC 0F                 0734            RRC                     MOVE THE HIGH FOUR DOWN
C2ED 0F                 0735            RRC
C2EE 0F                 0736            RRC
C2EF CD F3 C2           0737            CALL    HEOU1           PUT THEM OUT
C2F2 79                 0738            MOV     A,C             THIS TIME THE LOW FOUR
C2F3                    0739 *
C2F3 E6 0F              0740 HEOU1      ANI     0FH             FOUR ON THE FLOOR
C2F5 C6 30              0741            ADI     48              WE WORK WITH ASCII HERE
C2F7 FE 3A              0742            CPI     58              0-9?
C2F9 DA FE C2           0743            JC      OUTH            YUP!
C2FC C6 07              0744            ADI     7               MAKE IT A LETTER
C2FE 47                 0745 OUTH       MOV     B,A             OUTPUT IT FROM REGISTER 'B'
C2FF C3 4B C0           0746            JMP     VDMOT
C302                    0747 *
C302                    0748 *   COMPARE DE AND HL
C302                    0749 *
C302 7D                 0750 ACOMP      MOV     A,L
C303 93                 0751            SUB     E
C304 7C                 0752            MOV     A,H
C305 9A                 0753            SBB     D
C306 C9                 0754            RET     .               FLAGS ARE SET
C307                    0755 *
C307                    0756 *
C307                    0757 *          ENTER COMMAND
C307                    0758 *
C307                    0759 *  THIS ROUTINE GETS VALUES FROM THE KEYBOARD AND ENTERS
C307                    0760 * THEM INTO MEMORY.  THE INPUT VALUES ARE SCANNED FOLLOWING
C307                    0761 * A STANDARD 'GCLIN' INPUT SO ON SCREEN EDITING MAY TAKE
C307                    0762 * PLACE PRIOR TO THE LINE TERMINATOR.  A BACK SLASH '/'
C307                    0763 * ENDS THE ROUTINE AND RETURNS CONTROL TO THE COMMAND MODE.
C307                    0764 *
C307 CD 3E C2           0765 ENTER      CALL    SCONV           SCAN OVER CHARS AND GET ADDRESS
C30A E5                 0766            PUSH    H               SAVE ADDRESS
C30B                    0767 *
C30B CD 1A C2           0768 ENLOP      CALL    CRLF
C30E 06 3A              0769            MVI     B,':'
C310 CD CC C1           0770            CALL    CONT            GET LINE OF INPUT
C313 CD 3D C1           0771            CALL    CREM            REMOVE THE CURSOR
C316 0E 01              0772            MVI     C,1             START SCAN
C318 CD 25 C1           0773            CALL    VDAD2           GET ADDRESS
C31B EB                 0774            XCHG                    ....TO DE
C31C                    0775 *
C31C 0E 03              0776 ENLO1      MVI     C,3             NO MORE THAN THREE SPACES BETWEEN VALUES
C31E CD 34 C2           0777            CALL    SCHR1 SCAN TO NEXT VALUE
C321 CA 0B C3           0778            JZ      ENLOP           LAST ENTRY FOUND START NEW LINE
C324 1A                 0779 ENLO2      LDAX    D               GET THE CHR
C325 FE 2F              0780            CPI     '/'             COMMAND TERMINATOR?
C327 CA 04 C1           0781            JZ      COMND           IF SO...
C32A CD 44 C2           0782            CALL    SHEX            CONVERT VALUE
C32D 7D                 0783            MOV     A,L             GET LOW PART AS CONVERTED
C32E E1                 0784            POP     H               GET MEMORY ADDRESS
C32F 77                 0785            MOV     M,A             PUT IN THE VALUE
C330 23                 0786            INX     H
C331 E5                 0787            PUSH    H               BACK GOES THE ADDRESS
C332 C3 1C C3           0788            JMP     ENLO1           CONTINUE THE SCAN
C335                    0789 *
C335                    0790 *
C335                    0791 *
C335                    0792 *          EXECUTE COMMAND
C335                    0793 *
C335                    0794 *  THIS ROUTINE GETS THE FOLLOWING PARAMETER AND DOES A
C335                    0795 * PROGRAM JUMP TO THE LOCATION GIVEN BY IT.  IF PROPER
C335                    0796 * STACK OPERATIONS ARE USED WITHIN THE EXTERNAL PROGRAM
C335                    0797 * IT CAN DO A STANDARD 'RET'URN TO THE CONSOL COMMAND MODE.
C335                    0798 *
C335                    0799 *
C335 CD 3E C2           0800 EXEC       CALL    SCONV           SCAN PAST BLANKS AND GET PARAMETER
C338 E9                 0801            PCHL                    GO.....(AND TAKE NOTE)
C339                    0802 *
C339                    0803 *
C339                    0804 *
C339                    0805 *          TAPE LOAD COMMAND
C339                    0806 *
C339                    0807 *    THIS ROUTINE READS FROM EITHER TAPE UNIT PLACING
C339                    0808 * THE READ DATA INTO MEMORY.  WHILE SPACE WITHIN CONSOL
C339                    0809 * DOES NOT ALLOW FOR 'STANDARD' TAPE ROUTINES THIS
C339                    0810 * COMMAND WILL LOAD SOL- BASIC5 AND OTHER STANDARD SOL
C339                    0811 * SYSTEM SOLFTWARE FOR DIRECT EXECUTION.
C339                    0812 *
C339                    0813 *
C339 CD 24 C2           0814 TLOAD      CALL    SBLK            SCAN TO SPEED PARAMETER
C33C CA 4A C3           0815            JZ      DFLT            DEFAULT TO HIGH SPEED IF NONE
C33F CD 44 C2           0816            CALL    SHEX            CONVERT IT
C342 7D                 0817            MOV     A,L             GET VALUE
C343 E6 01              0818            ANI     1               ONLY BIT ZERO COUNTS
C345 3E 2d              0819            MVI     A,32            PRETEND ITS SLOW
C347 C2 4B C3           0820            JNZ     SETSP
C34A                    0821 *
C34A AF                 0822 DFLT       XRA     A               MAKE IT FAST
C34B F6 Cd              0823 SETSP      ORI     TAPE1+TAPE2     CONSOL STARTS BOTH TAPES
C34D D3 FA              0824            OUT     STAPT           START TAPES AND SELECT SPEED
C34F CD F1 C3           0825            CALL    DELAY           WAIT WHILE THE TAPE UNIT WINDS UP
C352 DB F8              0826            IN      TDATA           CLEAR THE UART FLAGS
C354                    0827 *
C354 CD 94 C3           0828 TLOD1      CALL    RHEAD           READ PAST HEADER
C357 C2 54 C3           0829            JNZ     TLOD1           IF ERROR START OVER
C35A                    0830 *
C35A 2A 0C C8           0831            LHLD    BLOCK           GET BLOCK SIZE
C35D EB                 0832            XCHG    .               ...TO DE
C35E 2A 0E C8           0833            LHLD    LOADR           GET LOAD ADDRESS
C361                    0834 *
C361 7A                 0835 LDLOOP     MOV     A,D             GET COUNT
C362 B3                 0836            ORA     E
C363 CA ED C3           0837            JZ      TOFF            COUNT IS ZERO-TURN OFF TAPE AND RETURN
C366 01 00 FF           0838            LXI     B,-256          THIS MANY PRIOR TO CRC TEST
C369 EB                 0839            XCHG    .               COUNT TO HL
C36A 09                 0840            DAD     B               A LITTLE MATH
C36B D2 E1 C3           0841            JNC     LBLK            NO CARRY, IT'S THE LAST BLOCK
C36E 06 00              0842            MVI     B,0             256 TO READ
C370                    0843 *
C370 0E 00              0844 RDBLK      MVI     C,0             ZERO THE CRC
C372 EB                 0845            XCHG    .               ROUND ROBIN
C373                    0846 *
C373 CD C6 C3           0847 RTBYT      CALL    TAPIN           GET CHARACTER
C376 77                 0848            MOV     M,A             STORE IT
C377 23                 0849            INX     H               BUMP MEMORY LOCATION
C378 A9                 0850            XRA     C               UPDATE THE CRC
C379 2F                 0851            CMA
C37A 91                 0852            SUB     C
C37B 4F                 0853            MOV     C,A             STORE THE NEW
C37C 05                 0854            DCR     B               COUNT DOWN
```

```
C37D C2 73 C3        0854          JNZ     RLBYT     STILL MORE IF NOT ZERO
C380                 0855 *
C380 CD C1 C3        0856          CALL    CRCCK     CHECK CRC AND FALL THROUGH TO ERROR IF NO GOOD
C383 CA 61 C3        0857          JZ      LOOLOOP   TEST OK
C386                 0858 *
C386 06 07           0859 TERR     MVI     B,'G'-40H BELL CHARACTER
C388 CD 48 C0        0860          CALL    VDOUT     PUT IT ON THE SCREEN
C38B C3 B4 C1        0861          JMP     COMND
C38E                 0862 *
C38E                 0863 *
C38E                 0864 *    CONSOL ERROR HANDLER
C38E                 0865 *
C38E EB              0866 ERR1     XCHG    .         GET SCAN ADDRESS
C38F 36 3F           0867 ERR2     MVI     M,'?'     PUT A QUESTION MARK THERE
C391 C3 B4 C1        0868          JMP     COMND     AND GO TO COMMAND MODE
C394                 0869 *
C394                 0870 *    READ THE HEADER
C394                 0871 *
C394 06 0A           0872 RHEAD    MVI     B,16      FIND 10 NULLS
C396 DB FA           0873 RHEA1    IN      STAPT     GET A BYTE
C398 E6 40           0874          ANI     TDR
C39A CA 96 C3        0875          JZ      RHLA1
C39D DB FB           0876          IN      TDATA     IGNORE ERROR CONDITIONS
C39F B7              0877          ORA     A         ZERO?
C3A0 C2 94 C3        0878          JNZ     RHLAD
C3A3 05              0879          DCR     B
C3A4 C2 96 C3        0880          JNZ     RHLA1     LOOP UNTIL 10 IN A ROW
C3A7                 0881 *
C3A7                 0882 *    WAIT FOR THE START CHARACTER
C3A7                 0883 *
C3A7 CD C6 C3        0884 SOHL     CALL    TAPIN
C3AA 3D              0885          DCR     A
C3AB C2 A7 C3        0886          JNZ     SOHL      WAIT FOR A '1'
C3AE                 0887 *
C3AE                 0888 *    NOW GET THE HEADER
C3AE                 0889 *
C3AE 21 05 C0        0890          LXI     H,THEAD   POINT TO BUFFER
C3B1 01 00 10        0891          LXI     B,HLEN*256  LENGTH OF HEADER IN 'B',C=0
C3B4                 0892 *
C3B4 CD C6 C3        0893 RHED1    CALL    TAPIN     GET BYTE
C3B7 77              0894          MOV     M,A       STORE IT
C3B8 23              0895          INX     H         INCREMENT ADDRESS
C3B9 A9              0896          XRA     C         NOW CALCULATE THE CRC
C3BA 2F              0897          CMA     .         INSIDE OUT AND UPSIDE DOWN
C3BB 91              0898          SUB     C         SQUEEZE IT
C3BC 4F              0899          MOV     C,A       AND SAVE AGAIN
C3BD 05              0900          DCR     B         WHOLE HEADER YET?
C3BE C2 B4 C3        0901          JNZ     RHED1     LOOP UNTIL DONE
C3C1                 0902 *
C3C1                 0903 *   THIS ROUTINE GETS THE NEXT BYTE AND COMPARES IT
C3C1                 0904 * TO THE VALUE IN REGISTER C.  THE FLAGS ARE SET ON
C3C1                 0905 * RETURN.
C3C1                 0906 *
C3C1 CD C6 C3        0907 CRCCK    CALL    TAPIN     GET CRC BYTE
C3C4 B9              0908          CMP     C         COMPARE IT WITH CALCULATED
C3C5 C9              0909          RET
C3C6                 0910 *
C3C6                 0911 *
C3C6                 0912 *   THIS ROUTINE GETS THE NEXT AVAILABLE BYTE FROM THE
C3C6                 0913 *  TAPE.  WHILE WAITING FOR THE BYTE THE KEYBOARD IS TESTED
C3C6                 0914 *  FOR A 'MODE' COMMAND.  IF RECEIVED THE TAPE LOAD IS
C3C6                 0915 *  TERMINATED AND A RETURN TO THE COMMAND MODE IS MADE.
C3C6                 0916 *
C3C6 DB FA           0917 TAPIN    IN      STAPT     CHECK STATUS
C3C8 E6 40           0918          ANI     TDR
C3CA C2 D7 C3        0919          JNZ     TRDY      ONE IS AVAILABLE
C3CD DB FC           0920          IN      KDATA     CHECK FOR MODE WHILE WE'RE WAITING
C3CF FE 80           0921          CPI     MODE
C3D1 CA D4 C1        0922          JZ      COMND     MODE WAS GIVEN..ABORT OPERATION
C3D4 C3 C6 C3        0923          JMP     TAPIN     NOT MODE...STAY IN LOOP
C3D7                 0924 *
C3D7 DB FA           0925 TRDY     IN      STAPT
C3D9 E6 18           0926          ANI     TFE+TOE   DATA ERROR?
C3DB C2 86 C3        0927          JNZ     TERR      IF FRAMING OR OVERRUN ERROR
C3DE DB FB           0928          IN      TDATA     GET THE DATA
C3E0 C9              0929          RET
C3E1                 0930 *
C3E1                 0931 * THIS ROUTINE CALCULATES THE LENGTH OF THE LAST BLOCK
C3E1                 0932 *
C3E1 01 FF FF        0933 LBLK     LXI     B,-1

C3E4 09              0934          DAD     B         COMPLEMENT HL
C3E5 23              0935          INX     H         .....TWO'S
C3E6 45              0936          MOV     B,L       LENGTH TO REGISTER B
C3E7 21 00 00        0937          LXI     H,0       TELL DE WE'RE DONE
C3EA C3 70 C3        0938          JMP     RDBLK     ONWARD TO THE END
C3ED                 0939 *
C3ED                 0940 *   THIS ROUTINE TURNS THE TAPE UNITS OFF
C3ED                 0941 *
C3ED AF              0942 TOFF     XRA     A
C3EE D3 FA           0943          OUT     STAPT     GIVE COMMAND
C3F0 C9              0944          RET               AND GRIND TO A SLOW STOP
C3F1                 0945 *
C3F1                 0946 *
C3F1 11 00 00        0947 DELAY    LXI     D,0       START LOOP
C3F4 1B              0948 DLOP1    DCX     D         DOWN COUNT
C3F5 7A              0949          MOV     A,D
C3F6 B3              0950          ORA     E         TEST FOR ZERO
C3F7 C2 F4 C3        0951          JNZ     DLOP1     IF NOT
C3FA C9              0952          RET
C3FB                 0953 *
C3FB                 0954 *
C3FB                 0955 *
C3FB                 0956 *
C3FB                 0957 *                 << SOL SYSTEM EQUATES >>
C3FB                 0958 *
C3FB                 0959 *
C3FB                 0960 *           DISPLAY PARAMETERS
C3FB                 0961 *
C3FB                 0962 VDMEM    EQU     0CC00H    SCREEN MEMORY
C3FB                 0963 *
C3FB                 0964 *
C3FB                 0965 *           KEYBOARD SPECIAL KEY ASSIGNMENTS
C3FB                 0966 *
C3FB                 0967 DOWN     EQU     9AH
C3FB                 0968 UP       EQU     97H
C3FB                 0969 LEFT     EQU     81H
C3FB                 0970 RIGHT    EQU     93H
C3FB                 0971 LOADK    EQU     8CH       LOAD KEY
C3FB                 0972 MODE     EQU     80H
C3FB                 0973 CLEAR    EQU     88H
C3FB                 0974 HOME     EQU     08EH
C3FB                 0975 BACKS    EQU     5FH       BACKSPACE
C3FB                 0976 LF       EQU     10
C3FB                 0977 CR       EQU     13
C3FB                 0978 BLANK    EQU     ' '
C3FB                 0979 SPACE    EQU     BLANK
C3FB                 0980 CX       EQU     'X'-40H
C3FB                 0981 *
C3FB                 0982 *           PORT ASSIGNMENTS
C3FB                 0983 *
C3FB                 0984 STAPT    EQU     0FAH      STATUS PORT GENERAL
C3FB                 0985 SERST    EQU     0F8H      SERIAL STATUS PORT

C3FB                 0986 SDATA    EQU     0F9H      SERIAL DATA
C3FB                 0987 TDATA    EQU     0FBH      TAPE DATA
C3FB                 0988 KDATA    EQU     0FCH      KEYBOARD DATA
C3FB                 0989 PDATA    EQU     0FDH      PARALLEL DATA
C3FB                 0990 DSTAT    EQU     0FEH      DISPLAY PARAMETER PORT
C3FB                 0991 SENSE    EQU     0FFH      SENSE SWITCHES
C3FB                 0992 *
C3FB                 0993 *
C3FB                 0994 *
C3FB                 0995 *           BIT ASSIGNMENT MASKS
C3FB                 0996 *
C3FB                 0997 SCD      EQU     1         SERIAL CARRIER DETECT
C3FB                 0998 SDSR     EQU     2         SERIAL DATA SET READY
C3FB                 0999 SPE      EQU     4         SERIAL PARITY ERROR
C3FB                 1000 SFE      EQU     8         SERIAL FRAMING ERROR
C3FB                 1001 SOE      EQU     16        SERIAL OVERRUN ERROR
C3FB                 1002 SCTS     EQU     32        SERIAL CLEAR TO SEND
C3FB                 1003 SDR      EQU     64        SERIAL DATA READY
C3FB                 1004 STBE     EQU     128       SERIAL TRANSMITTER BUFFER EMPTY
C3FB                 1005 *
C3FB                 1006 KDR      EQU     1         KEYBOARD DATA READY
C3FB                 1007 PDR      EQU     2         PARALLEL DATA READY
C3FB                 1008 PXDR     EQU     4         PARALLEL DEVICE READY
C3FB                 1009 TFE      EQU     8         TAPE FRAMING ERROR
C3FB                 1010 TOE      EQU     16        TAPE OVERFLOW ERROR
C3FB                 1011 TDR      EQU     64        TAPE DATA READY
C3FB                 1012 TTBE     EQU     128       TAPE TRANSMITTER BUFFER EMPTY
C3FB                 1013 *
C3FB                 1014 SOK      EQU     1         SCROLL OK FLAG
C3FB                 1015 *
C3FB                 1016 TAPE1    EQU     64        TAPE ONE 'ON' BIT
C3FB                 1017 TAPE2    EQU     128       TAPE TWO
C3FB                 1018 *
C3FB                 1019 *
C3FB                 1020 *
C3FB                 1021 *
C3FB                 1022 *                 SOL SYSTEM GLOBAL AREA
C3FB                 1023 *
C3FB                 1024          ORG     0C800H    START OF 1K RAM AREA
C800                 1025 *
C800                 1026 SYSRAM   EQU     $         START OF SYSTEM RAM
C800                 1027 SYSTP    EQU     $+1024    STACK IS AT THE TOP
C800                 1028 *
C800                 1029 *
C800                 1030 *           CONSOL PARAMETER AREA
C800                 1031 *
C800                 1032 NCHAR    DS      1         CURRENT CHARACTER POSITION
C801                 1033 LINE     DS      1         CURRENT LINE POSITION
C802                 1034 BOJ      DS      1         BEGINNING OF TEXT DISPLACEMENT
C803                 1035 OPORT    DS      1         OUTPUT PORT
C804                 1036 IPORT    DS      1         INPUT PORT
C805                 1037 *
C805                 1038 *
C805                 1039 *
C805                 1040 THEAD    DS      5         NAME
C80A                 1041          DS      1         THIS BYTE MUST BE ZERO
C80B                 1042 HTYPE    DS      1         TYPE
C80C                 1043 BLOCK    DS      2         BLOCK SIZE
C80E                 1044 LOADR    DS      2         LOAD ADDRESS
C810                 1045 XEQAD    DS      2         AUTO EXECUTE ADDRESS
C812                 1046 HSPR     DS      3         SPARES
C815                 1047 *
C815                 1048 HLEN     EQU     $-THEAD   LENGTH OF HEADER
C815                 1049 *
C815                 1050 *
```

LET'S HEAR FROM YOU

Your letters are the lifeblood for us

We really do appreciate your letters and comments about INTERFACE AGE. We need to know what you think about the publication so that we know in what ways we can make it better.

If you like it, tell us, and if there are things you think could be improved, tell us that, too.

We do read all our mail and we appreciate your feedback.

The Editors

LLL 8080 Basic Interpreter Program

PART II

By John Dickenson and Jerry Barber

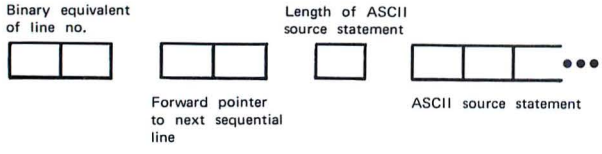
INTRODUCTION

This article is part #2 of a series of four articles covering the LLL 8080 BASIC Interpreter program released to the public domain by Lawrence Livermore Laboratories. This article covers the description of the BASIC Interpreter and includes the assembly listing of the LLL 8080 BASIC Interpreter program.

DESCRIPTION OF BASIC INTERPRETER

Following is a brief description of the BASIC interpreter. Hopefully, with this description, it will not be a major project to modify the BASIC to satisfy the reader's specific needs.

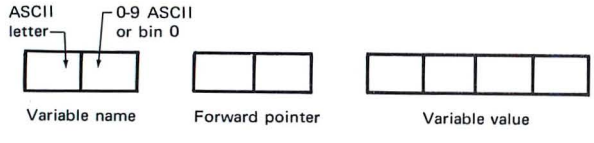
Formats — Source statements are stripped of blanks on input (character strings enclosed in " "s are an exception) and stored as is in memory, using the following format:



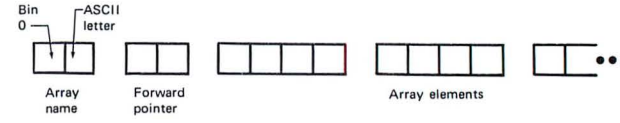
The forward pointer links statements by ascending line numbers. The last line's forward pointer (supposedly an end statement) has value 177777_8 to indicate end of the list.

The symbol table is built up at run time and begins after the most recently entered source statement (the variable STSPAC points to where the symbol table will start). Symbol table entries are shown below:

SCALAR-VARIABLE FORMAT



ARRAY-VARIABLE FORMAT



Subroutines — Following is a list of potentially useful subroutines, with a brief description of each subroutine:

ALPHA — Value pointed to by H and L is tested to see if it is an ASCII letter.
CY = 1 => Yes
CY = 0 => No

NUMB — Same as above but tests for a decimal number (ASCII 0-9).

CHAR2 — Inputs a character from the teletype to a register.

CHAR5 — Same as above for HSR (High Speed Paper Tape Reader).

CHK1 — Checks to see if HL are equal to 177777_8 (−1).
CY = 1 => Yes
CY = 0 => No.

CONV (CVRT) — One of the floating-point routines. Converts floating-point number to a character string. Output is padded to the output buffer.

COPDH — Copies floating-point number pointed to by D, E to location pointed to by H, L; uses copy.

COPY — One of the floating-point routines. Copies floating-point value pointed to by A, L to location pointed to by H, C.

CUB — Converts the integer-character string pointed to by H, L to its binary equivalent. Vale returns in D, E registers.

DCOMP — Double-byte comparison routine. Compares value in CB to the value in ED.

Z = 1 => CB = ED
CY = 1 => CB > ED
CY = 0 => CB < ED.

DFXL — One of the floating-point routines. Used to float an unsigned integer H, L point to first of four bytes; integer is right justified in first three bytes.

EVAL — Evaluates an expression the first element of which is pointed to by H, L and the length of which is in C. Used to evaluate expressions wherever they are legal in BASIC. C usually contains the length of the source statement line containing the expression.

FINPT — One of the floating-point routines. Converts character string to floating-point number. The variable HLINP contains a pointer to the character string, and the variable CREG contains the length of line containing character string. Mode = 0 => data comes from teletype (i.e., only delimiters are g's). Mode = 1 => data comes from source statements.

FIX — Fixes a floating-point number. DE points to number to be fixed. Error code 13 is given if number is too big to fix.

FSYM — Finds symbols in symbol table. BC contains symbol. Returns with HL pointing to symbol value.
CY = 1 => symbol was found.
CY = 0 and a scalar => symbol not found, but inserted and initialized to 0.
CY = 0 and an array => not found, no action taken: HL are meaningless.

LADD — Floating-point add routine.

LSUB — Floating-point subtract routine.

LOIU — Floating-point divide routine.

LMUL — Floating-point multiply routine.

LMCM — One of the floating-point routines. Compares two floating-point values, HL Point to first floating-point values and HB point to second floating-point value.
z =1 => Equality
Cy=1 => first < second
(Note: compares absolute only, does not reference mantissa sign.)

MCHK — Waits for flag from port 3. Proper mask is sent in register B.

MEMFUL — Checks to see if memory is full. HL point to location of memory to be checked. Memory is considered full if it is within 50_{10} locations of the current value of stack pointer.

MULT — Multiplies two two-byte binary numbers. HL point to last byte of four bytes. First two contain first number. Last two contain second number. Answer returns in BCDE.

NSRCH — Routine to locate source line in memory passed binary value of line number in DE. Returns address of line in HL, CY=1 => not found.

OUTR — Used by CONV (CURT) to pad output to output buffer.

PAD — Pads characters to output buffer. A contains character; B contains number of pads.

SYMSRT — Checks a character string to see if it is a BASIC symbol. HL contains address pointing to 1st character of symbol, C contains length of line that contains symbol. A contains type of symbol sought.
0=command 1=keyword
z=operator or delimiter 3=function

Returns with 377_8 in a register if nothing found. Otherwise A contains symbol number in appropriate KDAT table. Thus, for symbol type 2, if a 4 is returned, the symbol found was the fourth one (starting with 0) in table KDAT3 (KDAT concatenated with 2 and 1 or A'). CIS is updated, but HL is not.

TTYIW — Inputs a line from teletype. Stores starting address at location pointed to by HL. Line edits. Returns length of line in A register (maximum line length is 72 characters).

VALUE — Called with HL pointing to A variable, constant, or function; C contains line length, returns with DE pointing to floating-point value. HL, C are updated.

VAR — Called with HL pointing to character string, C has line length. Determines if character string is a variable. If so, returns with CY=1, DE pointing to value (subscripts of arrays are evaluated, etc.). HL, C updated. If not, a variable returns CY=0, HL, C untouched.

WRIT — Dumps contents of output buffer to teletype. Uses entry WRIT1 with D register equal to one to suppress CR/LF.

ZROL — Part of floating-point subroutines. Writes a floating-point zero, starting at location pointed to by HL.

The preceding list contains those subroutines most likely to be used by someone modifying BASIC. If you plan on using one of the routines, you should examine it and its comments carefully.

Variables — Following is a list of interpreter variables, with a description of each variable:

MEMST — Assembly time variable. Contains the first available RAM location. This is where active variables start.

MEMEND — Assembly time variable. Contains the last available location in RAM.

SEND — Has value 6, used with RST instruction to print characters via ODT.

OBUFF — Output buffer, the first location contains the number of characters in the buffer + 1.

IBUF — Input buffer, occupies same area as OBUFF.

STLINE — Points to first source line to be executed. If no source, contains 177777_8.

NLINE, NL2, NL4, NL6 — Contain address, binary-equivalent line number, forward pointer, and length of next input line.

KLINE, KL2, KL4, KL6 — Same as above, but used by a subroutine that inserts lines in sequential order (insert).

PLINE, PL2, PL4, PL6 — Subroutine insert to order statements sequentially.

KASE, LEN — Temporary storage for command mode routines.

MULT1, MULT2 — Used to store binary values to be multiplied.

SBSAV — Temporary storage for call-statement processor.

STSPAC — Next available location in memory, symbol table starts here at run time.

LPNT — Pointer to the current line at run time.

CPNT — Pointer to current character in current line at run time.

KFPNT — Point to next sequential line at run time.

FREG1, FREG2 — Two floating-point registers.

HLINP, CREG — Temporary storage for HL and C registers for routine INP.

NXTSP — Pointer to next available space of memory for symbol table.

GREG — General register, in and out instructions are stored here and executed for get and put functions.

MODE — Indicates to INP routine whether input data comes from source or teletype.

MESCR — Temporary storage for call-statement processor. Points to next available space after symbol table. Area after the symbol table is used to store intermediate results of expressions or constants passed to user subroutines.

VARAD — Temporary storage space for input-statement processor.

VEND — Assembly time variable. Indicates end of interpreter variable-storage area and where FWAM pointer is to go.

FWAM — First word of available memory pointer. This is where user source programs go.

Some of the above variables occupy the same area of memory. This is because some variables are used only in the command mode and others only at runtime. To conserve space, they share the same memory locations.

New BASIC Statements — To add additional statements to the BASIC, use the following procedure. First, insert the statement keyword in the data tables for subroutine SYMSRT. Then, insert the starting address of the statement processor in the interpreter JUMP table. Finally, the statement processor itself must be inserted.

The keyword must be entered in the table KDAT2. The first byte must be the keyword length and the next bytes hold the ASCII-coded keyword. The table must end with A 377_8. If the keyword is the Nth entry in the table, on return from SYMSRT, the A register will hold N-1 if the keyword is found.

The starting address of the statement processor must be inserted into table JTBL. The order of keywords in KDAT2 must correspond with statement processor addresses in JTBL since, on return from SYMSRT, the A register times two is used as offset in JTBL to determine processor address.

The statement processor must be placed somewhere in memory. Generally, the first thing done in the statement processors is to load the pointer to the statement (LHLD CPNT) and increment past the keyword (since HL is not updated by SYMSRT). On entry, C contains the number of characters in the line minus those checked by SYMSRT. The end of the processor should be a "JMPIEND" instruction.

New Functions — New functions must be added to SYSSRT Data Table KDAT4 in the same manner as for key words. The function itself must be placed in subroutine "VALUE." Presently, the only function in VALUE is GET.

Message Lines — The following description tells how to incorporate messages into BASIC output routines. Currently, to output a message to the teletype, the user executes an LXI H,ODATA, then a call to FORMK where K is an integer indicating which message is wanted (i.e., K=z indicates "TURN ON PUNCH"). FORM pads the message into the output buffer. Then A "CALL WRIT" writes the contents of the buffer.

Suppose the message "POTATO BASIC" is to be added. Preceding the form 9 instruction, we will insert "FOR10: INR L." At the end of the ODATA table, we

add "DB ODAT8 and 377Q.". And, after message ODAT 7, we add ODAT8 DB *12, "POTATO BASIC." Now, the following program segment:

```
LXI   H, ODATA
CALL  FOR10
CALL  WRIT,
```

will cause "POTATO BASIC" to be output to the teletype.

SEE MICROCOMPUTER SOFTWARE DEPOSITORY PROGRAM INDEX FOR COPIES OF THIS PROGRAM

```
8080 MACRO ASSEMBLER, VER 2.1  ERRORS = 0

                        ;
                        ;       BAS80 - BASIC INTERPRETER FOR INTEL 8080 MICRO-PR
                        ;
                        ;         WRITTEN AT THE UNIVERSITY OF IDAHO BY:
                        ;
                        ;           GERALD R. BARBER
                        ;           DEPT. OF ELECTRICAL ENGINEERING AND COMPUTER
                        ;           MASSACHUSETTES INSTITUTE OF TECHNOLOGY
                        ;           CAMBRIDGE, MASSACHUTTES
                        ;
                        ;           JOHN A. TEETER
                        ;           EAST OF CASCADE
                        ;           SOUTH OF LANDMARK
                        ;           DEARBORN, IDAHO
                        ;
                        ;           JOHN W. ERICKINSON
                        ;           DEPT. OF ELECTRICAL ENGINEERING
                        ;           UNIVERSITY OF IDAHO
                        ;           MOSCOW, IDAHO
                        ;
                        ;       THIS FILE CONTAINS THE EDIT AND EXECUTION PORTION
                        ;       OF THE BASIC INTERPRETER.  IT IS DESIGNED TO EXEC
                        ;       WITH THE COMPANION FILE CONTAINING THE FLOATING
                        ;       POINT ROUTINES.
                        ;
                        ;       I/O IS ACCOMPLISHED WITH A TTY AND A HIGH SPEED
                        ;       PAPER TAPE READER.  TTY I/O IS DONE THROUGH
                        ;       OUT ROUTINES.  PAPER TAPE INPUT IS HANDLED
                        ;       MANUALY.
                        ;
                        ;       DEFINE LINKAGES FROM FLT PNT PACKAGE TO I/O ROUTINES
                        ;
0FEA                            ORG     77720
0FEA  C3 73 19                  JMP     INP
0FED  C3 BA 17                  JMP     OUTR
                        ;
                        ;       DEFINE ADDRESSES OF ACTIVE VARIABLES
                        ;
2100                    MEMST   EQU     20400Q          ;MUST BE ON PAGE BOUNDA
27FF                    MEMEND  EQU     23777Q          ;END OF RAM MEMORY
0006                    SEND    EQU     6               ;RST FOR ODT
2100                    OBUFF   EQU     MEMST           ;INPUT AND OUTPUT BUFFE
2101                    IBUF    EQU     MEMST+1         ;SAME AREA
2149                    STLINE  EQU     MEMST+111Q      ;FIRST LINE OF USER PRO
214B                    NLINE   EQU     MEMST+113Q      ;POINTER TO NEW LINE OF
2140                    NL2     EQU     MEMST+115Q      ;BINARY VALUE OF NEW LI
214F                    NL4     EQU     MEMST+117Q      ;FORWARD POINTER OF NEW
                        ;
2151                    NL6     EQU     MEMST+121Q      ;LENGTH OF NEW LINE
2152                    KLINE   EQU     MEMST+122Q      ;POINTER TO CURRENT LIN
2154                    KL2     EQU     MEMST+124Q      ;BINARY EQU. OF CURRENT
2156                    KL4     EQU     MEMST+126Q      ;FORWARD POINTER OF CUR
2158                    KL6     EQU     MEMST+130Q      ;LENGTH OF NEW LINE
2159                    PLINE   EQU     MEMST+131Q      ;PREVIOUS LINE POINTER-
215B                    PL2     EQU     MEMST+133Q      ;BINARY LINE NUMBER OF
215D                    PL4     EQU     MEMST+135Q      ;FORWARD POINTER OF PRE
215D                    SBSAV   EQU     PL4             ;RETURN ADD. SAVE FOR C
215F                    PL6     EQU     MEMST+137Q      ;LENGTH OF PREVIOUS LIN
2160                    KASE    EQU     MEMST+140Q      ;SPARE STORAGE USED AS
2161                    LFN     EQU     MEMST+141Q      ;ALSO SPARE-OFTEN LENGT
2162                    MULTI   EQU     MEMST+142Q      ;FIRST OF TWO WORDS USE
2164                    MULT2   EQU     MEMST+144Q      ;SECOND WORD
2159                    NXTSP   EQU     PLINE           ;POINTER TO NEXT AVAILA
2148                    STSPAC  EQU     NLINE           ;START OF SYM. TABLE
2ECD                    ZKOL    EQU     7715Q           ;FLOATING PNT. ROUTINE
2152                    LPNT    EQU     KLINE           ;SOURCE LINE PTR DURING
215A                    KLEN    EQU     KL6             ;SPARE-USED AS NEEDED
215B                    CPNT    EQU     PL2             ;CHAR. PTR DURING EXEC.
215F                    KFPNT   EQU     PL6             ;FWD LINE PTR DURING EX
2180                    FREG2   EQU     MEMST+200Q      ;FLOATING POINT REG. #2
2184                    CREG    EQU     MEMST+204Q      ;TEMP. SPACE FOR ROUTIN
0FD0                    LAUP    EQU     77200           ;FLOATING POINT ROUTINE
0FD3                    LMUL    EQU     7723J           ;FLOATING POINT ROUTINE
0FD6                    LDIV    EQU     77266           ;FLOATING POINT ROUTINE
0FD9                    LSUB    EQU     7731J           ;FLOATING POINT ROUTINE
0FDC                    LFXL    EQU     7734J           ;FLOATING POINT ROUTINE
0FDF                    LMCM    EQU     77377           ;FLOATING POINT ROUTINE
2186                    HLINP   EQU     MEMST+206Q      ;TEMP. SPACE FOR ROUTIN
2177                    GREG    EQU     MEMST+167Q      ;GENERAL PURPOSE REGIST
217C                    FREG1   EQU     MEMST+174Q      ;FLOATING PNT. REGISTER
2106                    SCR     EQU     MEMST+146Q      ;SCRATCH AREA FOR I/O R
0FE5                    CONV    EQU     77459Q          ;FLOATING PNT. OUTPUT R
2185                    MODF    EQU     MEMST+205Q      ;MODE FLAG FOR ROUTINE
0FE8                    FINPT   EQU     7750Q           ;FLOATING PNT. INPUT RO
0FEB                    MULT    EQU     77530           ;ADD, VALUE MULT. ROUTI
0FEE                    PTVAL   EQU     7756J           ;DEFINE LPNT,KFPNT ETC.
0FF1                    DCOMP   EQU     7761Q           ;DOUBLE BYTE COMPARISON
0FF4                    MCHK    EQU     77640           ;CHECK MASK
0FF7                    CHAR2   EQU     77677           ;INPUT CHAR. FROM TTY
2188                    MESCR   EQU     MEMST+210Q      ;DEFINE MEMORY SCR AREA
218A                    VARAD   EQU     MEMST+212Q      ;TEMP SPACE FOR INP. ST
218C                    VNAME   EQU     MEMST+214Q      ;TEMP SPACE FOR 'FOR-NE
218E                    VLOC    EQU     MEMST+216Q      ;TEMP SPACE FOR 'FOR-NE
2190                    FLIMT   EQU     MEMST+220Q      ;TEMP SPACE FOR 'FOR-NE
2194                    NEST    EQU     MEMST+224Q      ;NESTING STACK-POINTER
2196                    STAC    EQU     MEMST+226Q      ;FOR-NEXT NESTING STACK
3014                    STSIZ   SET     20              ;STACK SIZE, ALLOWS 10
219A                    TOPNS   EQU     STAC            ;TOP OF STACK
21AA                    BUTNS   EQU     STAC+STSIZ      ;BOTTOM OF STACK
                        ;
21AA                    VEND    EQU     MEMST+252Q      ;DEF. END OF VAR. STORA
                        ;
                        ;       DEFINE FWA MEMORY AND START OF SUBTABLE
                        ;
21AA                            ORG     VEND
21AA  AD 21             FWAM:   DW      SCMEM
21AC  FF                SUBS:   DB      377Q            ;NOTHING IN SUB TABLE
21AD                    SCMEM   EQU     $
                        ;
                        ;       MAIN ROUTINE--HANDLES ALL USER INPUT
                        ;
1000                            ORG     10000Q
                        ;
1000  21 00 21            M1:   LXI     H,OBUFF         ;INITIALIZE THE OUTPUT
1003  36 01                   MVI     M,1             ;TO HAVE LENGTH ONE.
1005  21 49 21                LXI     H,STLINE        ;PUT -1 INTO STLINE
1008  36 FF                   MVI     M,377Q
100A  2C                      INR     L
100B  36 FF                   MVI     M,377Q          ;-1 IS ALWAYS END OF LI
100D  2A AA 21                LHLD    FWAM            ;GET ADDRES OF FWA MEM.
1010  22 4B 21                SHLD    NLINE           ;STORE IN FREE SPACE PN
1013  31 F5 27            M1A:  LXI     SP,MEMEND-10    ;SET STACK TO HIGH MEMO
1016  21 5A 14            M2:   LXI     H,ODATA         ;PRINT TO ODATA (ASCII
1019  CD 48 14                  CALL    FORM1           ;PAC "READY"
101C  CD D0 12                  CALL    WRIT
101F  2A 4B 21            M3:   LHLD    NLINE           ;GET NEXT LINE FOR ACCE
1022  23                      INX     H
1023  23                      INX     H               ;MOVE PAST POINTERS TO
1024  23                      INX     H
1025  23                      INX     H
1026  23                      INX     H
1027  CD C9 13                CALL    TTYIN           ;GET INPUT FROM TTY
102A  4F                      MOV     C,A             ;MOVE NUMBER OF CHAR TO
102B  FE 00                   CPI     0               ;IS IT 0
102D  CA 1F 10                JZ      M3              ;YEP SO TRY AGAIN
1030  CD 22 12                CALL    ALPHA           ;IS INPUT AN ALPHA STRI
1033  DA 42 10                JC      M4              ;YEP--GO DECIPHER
1036  CD 14 12                CALL    NUMB            ;IS INPUT A NUMERIC STR
1039  D4 9E 11                CNC     WHAT            ;NOPE -- FIND OUT WHATS
103C  CD 8F 10                CALL    INSERT          ;YEP-- GO INSERT NEW LI
103F  C3 1F 10                JMP     M3              ;LOOP ON PROCESS
1042  3E 00            M4:   MVI     A,0             ;SET UP SYMBOL TYPE A=0
1044  CD F7 12                CALL    SYMSRT          ;GO FIND COMMAND
1047  3C            M4A:    INR     A
1048  CC 9E 11                CZ      WHAT            ;REG. A RETURNED WITH -
104B  3D                      DCR     A
104C  CA 78 16                JZ      RUN             ;ITS A RUN COMMAND
104F  3D                      DCR     A
1050  CC AA 11                CZ      TAPE            ;THE DUDE WANTS A PLST
1053  CA 16 10                JZ      M2
1056  3D                      DCR     A
1057  CC D1 11                CZ      LIST            ;HE WANTS A LISTING
105A  CA 16 10                JZ      M2
105D  3D                      DCR     A
105E  CA 00 10                JZ      M1              ;IT WAS A PURGE COMMAND
1061  3D                      DCR     A
1062  C4 9E 11                CNZ     WHAT            ;IF ITS NOT ZERO--THEN
                                                      ;ELSE ITS PTAPE
                        ;
                        ;       ROUTINE TO INPUT FROM HSR
                        ;       THIS CONDITION IS DEFAULT NOW
                        ;
1065  CD 08 15            PTAPE:  CALL    CHARS           ;GET CHAR. FROM READER
1068  FE 00                   CPI     0               ;IS IT LEADER?
106A  CA 65 10                JZ      PTAPE           ;YEP -- WAIT FOR GOOD I
106D  CD D5 14            PT1:  CALL    HSRIN           ;GOT CHAR.
1070  4F                      MOV     C,A
1071  FE 00                   CPI     0               ;IS IT STILL LEADER?
1073  CA 65 10                JZ      PTAPE           ;YEP-- KEEP WAITING
1076  CD 22 12                CALL    ALPHA           ;IS IT ALPHA STRING?
1079  DA A2 10                JC      M4              ;YEP-- GO GO INTERPRET
107C  CD 8F 10                CALL    INSERT          ;NOPE--GO INSERT IT
107F  CD 08 15                CALL    CHARS           ;GET NEXT CHAR.
1082  FE 00                   CPI     0               ;IS IT LEADER?
1084  CA 16 10                JZ      M2              ;YEP--GO BACK TO KEYBOA
1087  23                      INX     H
1088  23                      INX     H               ;MOVE PAST POINTERS OF
1089  23                      INX     H
108A  23                      INX     H
108B  23                      INX     H
108C  C3 6D 10                JMP     PT1             ;LOOP ON PROCESS
                        ;
                        ;       ROUTINE TO HANDLE ALL SOURCE LINE INPUT.
                        ;       THIS INCLUDES INSERTION, DELEATION, AND
                        ;       ADDITION OF NEW SOURCE LINES.
                        ;
108F  2B                INSERT: DCX     H               ;THE ROUTINE SORTS LINI
1090  71                      MOV     M,C             ;THE TEXT USING THE FOR
1091  23                      INX     H               ;STORED IN THE TABLE.
1092  CD 2A 12                CALL    CVB             ;GET BINARY FOR NEW LIN
1095  FE 05                   CPI     5               ;IS IT MIN. VALUE?
1097  DA A2 10                JC      ISR1A           ;ITS A VALID NUMBER
109A  C4 9E 11                CNZ     WHAT            ;INVALID NUMBER--REPORT
109D  7B                      MOV     A,E
109E  17                      RAL
109F  DC 9E 11                CC      WHAT
10A2  2A 4B 21            ISR1A:LHLD    NLINE           ;POINT TO NEW LINE
10A5  72                      MOV     M,D
10A6  23                      INX     H
10A7  73                      MOV     M,E
10A8  21 4B 21                LXI     H,NLINE
10AB  CD EE 0F                CALL    PTVAL           ;FIND THE POINTER VALUE
10AE  2A 49 21                LHLD    STLINE          ;GET START OF SOURCE
10B1  CD A3 12                CALL    CHR1            ;CHECK FOR NO TEXT
10B4  D2 BD 10                JNC     ISRT3           ;YEP--IT IS SOMETHING-STA
10B7  2A 4B 21                LHLD    NLINE           ;THIS IS FIRST LINE
10BA  22 49 21                SHLD    STLINE          ;MOVE NLINE TO STLINE
10BD  16 FF                   MVI     D,377Q          ;SET UP -1 FOR END OF L
10BF  5A                      MOV     E,D
10C0  CD 8F 11                CALL    STPNT           ;STORE IT AWAY IN POINT
10C3  23                      INX     H
10C4  7E                ISRT2:MOV     A,M             ;GETLINE LENGTH
10C5  C6 05                   ADI     5               ;ADJUST TO LEN+5
10C7  2A 4B 21                LHLD    NLINE
10CA  85                      ADD     L
10CB  6F                      MOV     L,A
10CC  3E 00                   MVI     A,0
10CE  8C                      ADC     H
10CF  67                      MOV     H,A
10D0  22 4B 21                SHLD    NLINC           ;STEP PAST NEW LINE
10D3  C9                      RET
10D4  22 52 21            ISRT3:SHLD    KLINE           ;SET UP CURRENT LINE
10D7  21 52 21            ISRT4:LXI     H,KLINE         ;GET CURRENT LINE AND
10DA  CD EE 0F                CALL    PTVAL           ;SET UP POINTERS
10DD  21 4D 21                LXI     H,NL2           ;GET BINARY VALUE
10E0  56                      MOV     D,M             ;OF NEW LINE
10E1  2C                      INR     L
10E2  5F                      MOV     E,M
10E3  21 54 21                LXI     H,KL2           ;GET CURRENT LINE BINAR
10E6  46                      MOV     B,M
10E7  2C                      INR     L
10E8  4E                      MOV     C,M
10E9  CD F1 0F                CALL    DCOMP           ;HOW DO THEY COMPARE?
10EC  CA 21 11                JZ      ISRT6           ;REPLACE/DELETE
10EF  DA 7A 11                JC      ISR12           ;KLINE>NLINE
10F2  2A 56 21                LHLD    KL4             ;MOVE ON TO NEXT LINE
10F5  CD A3 12                CALL    CHR1            ;ARE WE DONE?
10F8  DA 0F 11                JC      ISRT5
10FB  E5                      PUSH    H
10FC  2A 52 21                LHLD    KLINE
10FF  22 59 21                SHLD    PLINE           ;SET UP PREVIOUS LINE
1102  21 59 21                LXI     H,PLINE
1105  CD EE 0F                CALL    PTVAL           ;SET POINTERS
1108  E1                      POP     H
1109  22 52 21                SHLD    KLINE           ;SET UP CURRENT LINE
110C  C3 D7 10                JMP     ISRT4           ;LOOP ON PROCESS
110F  2A 4B 21            ISRT5:LHLD    NLINE           ;GET NLINE ADDRESS
1112  CD 91 11                CALL    NOLINE          ;IS IT 0 LEN. LINE?
1115  C8                      RZ                      ;YES-DO NOTHING
1116  EB                      XCHG
1117  2A 52 21                LHLD    KLINE           ;GET CURRENT LINE
111A  CD 8B 11                CALL    STPNT           ;STORE AWAY POINTERS
111D  EB                      XCHG
111E  C3 BD 10                JMP     ISRT1           ;GO SET END OF LIST
1121  2A 4B 21            ISRT6:LHLD    NLINE           ;REPLACE OR DELETE
1124  CD 91 11                CALL    NOLINE          ;NLINE MEANS DELETE
1127  C2 BB 11                JNZ     ISRT8           ;GO REPLACE IF NOT 0
112A  2A 49 21                LHLD    STLINE          ;GET STLINE
112D  EB                      XCHG
112E  2A 52 21                LHLD    KLINE           ;IF STLINE=KLINE THEN L
1131  E5                      PUSH    H               ;NOW ZERO
1132  C1                      POP     B
1133  CD F1 0F                CALL    DCOMP
1136  2A 56 21                LHLD    KL4
1139  CA 44 11                JZ      ISRT7           ;IF NOT STLINE THEN GO
113C  EB                      XCHG
113D  2A 59 21                LHLD    PLINE
1140  CD 8B 11                CALL    STPNT
1143  C9                      RET
1144  22 49 21            ISRT7:SHLD    STLINE
1147  C9                      RET
1148  2A 56 21            ISRT8:LHLD    KL4             ;REPLACE LINE
114B  EB                      XCHG
114C  2A 4B 21                LHLD    NLINE
114F  CD 8B 11                CALL    STPNT
1152  2A 52 21            ISRT9:LHLD    KLINE
1155  EB                      XCHG
1156  2A 49 21                LHLD    STLINE
1159  E5                      PUSH    H
115A  C1                      POP     B
115B  CD F1 0F                CALL    DCOMP
115E  CA 71 11                JZ      ISR11           ;IF KLINE =STLINE GOTO
1161  2A 4B 21                LHLD    NLINE
1164  EB                      XCHG
1165  2A 59 21                LHLD    PLINE
1168  CD 8B 11                CALL    STPNT           ;PREVIOUS LINE PNTS. TO
116B  21 51 21            ISR10:LXI     H,NL6           ;SET UP NEW LINE TO
116E  C3 C4 10                JMP     ISRT2           ;POINT TO M(KL4)
1171  2A 4B 21            ISR11:LHLD    NLINE           ;NLINE=STLINE
1174  22 49 21                SHLD    STLINE
1177  C3 6B 11                JMP     ISR10
117A  2A 52 21            ISR12:LHLD    KLINE           ;NLINE IN FRONT OF KLIN
117D  EB                      XCHG
117E  2A 4B 21                LHLD    NLINE
1181  CD 91 11                CALL    NOLINE
1184  C8                      RZ
1185  CD 8B 11                CALL    STPNT
1188  C3 52 11                JMP     ISRT9
                        ;
                        ;       ROUTINE TO STORE POINTERS INTO MEM ARRAY
                        ;       HL CONTAIN ADDR. OF FIRST WORD OF ENTRY.
                        ;       THE CORESPONDING ELEMENTS (HL2,HL4,HL6) CONTAIN
                        ;       THE NEW ENTRIES.
                        ;
118B  23                STPNT:  INX     H
118C  23                        INX     H
118D  73                        MOV     M,E             ;MOVE HL2/NL2/PL2 INTO
118E  23                        INX     H
118F  72                        MOV     M,D
1190  C9                        RET
                        ;
                        ;       ROUTINE TO CHECK NEW LINE FOR SOURCE STMT.
                        ;
1191  E5                NOLINE: PUSH    H
1192  23                        INX     H
1193  23                        INX     H
1194  23                        INX     H
1195  23                        INX     H
1196  4E                        MOV     C,M             ;MOVE LENGTH OF LINE IN
1197  23                        INX     H
1198  CD 71 12                  CALL    LENGTH          ;POINT TO FIRST CHAR.
119B  E1                        POP     H
119C  B9                        CMP     C               ;COMPARE LEN TO 0
119D  C9                        RET
```

```
                        ;
                        ; ROUTINE TO RESPOND WITH 'WHAT?' FOR UNIDENTIFIED
                        ; COMMAND.
                        ;
119E  21 5A 14          WHAT:   LXI     H,3DATA         ;GET OUTPUT BUFFER ADDR
11A1  CD 44 14                  CALL    FORM7           ;PAD IT
11A4  CD D0 12                  CALL    WRIT            ;DUMP IT
11A7  C3 13 10                  JMP     M1A             ;JUMP TO RESTART.
                        ;
                        ; ROUTINE TO PUNCH PAPER TAPE OF SOURCE.
                        ; CALLED IN RESPONSE TO TAPE COMMAND
                        ;
11AA  F5                TAPE:   PUSH    PSW
11AB  C5                        PUSH    B
11AC  21 5A 14                  LXI     H,3DATA         ;GET OUTPUT BUFFER ADDR
11AF  CD 47 14                  CALL    FORM2           ;PAD IN TURN ON PUNCH
11B2  CD D0 12                  CALL    WRIT            ;DUMP IT
11B5  3E 00                     MVI     A,0             ;SET UP TO DUMP LEADER
11B7  C1                        POP     B
11B8  06 40                     MVI     B,100Q
11BA  F5                        PUSH    PSW
11BB  C5                        PUSH    B
11BC  CD AD 12                  CALL    PAD             ;PAD 100 0'S
11BF  CD D0 12                  CALL    WRIT
11C2  C1                        POP     B
11C3  C5                        PUSH    B
11C4  CD D1 11                  CALL    LIST            ;GO DO STANDARD LIST
11C7  C1                        POP     B
11C8  F1                        POP     PSW
11C9  CD AD 12                  CALL    PAD             ;GO PAD AND DUMP TRAILE
11CC  CD D0 12                  CALL    WRIT
11CF  F1                        POP     PSW
11D0  C9                        RET
                        ;
                        ; ROUTINE TO LIST TO TTY THE SOURCE STMTS.
                        ; ALSO USED TO DUMP TAPE.
                        ; ALLOWS THE USER TO INPUT FIRST LINE OF
                        ; OF FIRST AND LAST LINES.
                        ;
11D1  2A 49 21          LIST:   LHLD    STLINE
11D4  CD A3 12                  CALL    CHK1            ;CHECK FOR NO SOURCE
11D7  DA 13 10                  JC      M1A             ;GO RESTART
11DA  22 59 21                  SHLD    PLINE           ;SET UP TO START LOOKIN
11DD  21 FF FF                  LXI     H,177777Q
11E0  22 52 21                  SHLD    KLINE
11E3  0D                        DCR     C
11E4  C4 40 15                  CNZ     BOUND           ;GO FIND FIRST LINE
11E7  2A 59 21                  LHLD    PLINE           ;STORE IT AWAY
11EA  23                LIS1:   INX     H
11EB  23                        INX     H
11EC  46                        MOV     B,M
11ED  23                        INX     H               ;GET LEN. IN REG B
11EE  4E                        MOV     C,M
11EF  C5                        PUSH    B
11F0  23                        INX     H
11F1  CD 49 14                  CALL    FORM5           ;GO PAD LINE
11F4  CD D0 12                  CALL    WRIT            ;GO DUMP LINE
11F7  C1                        POP     B
11F8  2A 52 21                  LHLD    KLINE           ;CHECK FOR LAST LINE
11FB  EB                        XCHG
11FC  CD F1 0F                  CALL    DCOMP
11FF  CA                        PZ
1200  68                        MOV     L,B
1201  61                        MOV     H,C
1202  CD 0B 12                  CALL    QUITT           ;CHECK FOR INTERRUPTION
1205  C3 EA 11                  JMP     LIS1            ;NONE - CONTINUE
                        ;
                        ; THIS ROUTINE CHECKS PORT 2 FOR A CNTRL/S CHARACTER
                        ; IF ONE IS FOUND THEN EXECUTION IS TO BE INTERRUPTED
                        ; CONTROL IS PASSED TO M1A
                        ;
1208  DB 03             QUITT:  IN      3               ;TEST FLAG PORT
120A  1F                        RAR                     ;FLAG TO CY
120B  D0                        RNC                     ;NOTHING THERE
120C  DB 02                     IN      2               ;FLAG WAS SET, GET DATA
120E  FE 93             QTCHK:  CPI     223Q            ;WAS IT CNTRL/S?
1210  CA 13 10                  JZ      M1A             ;YES
1213  C9                        RET                     ;NO, RETURN
                        ;
                        ; ROUTINES NUMB AND ALPHA CHECK IN CONTENTS OF MEMORY
                        ; LOCATION IN HL CONTAIN ASCII NUMERIC OR ALPHABETIC
                        ; CHARACTER. RETURN CY=1 IF YES, CY=0 IF NO.
                        ;
1214  C5                NUMB:   PUSH    B
1215  06 BD                     MVI     B,260Q          ;SET UP ASCII 0
1217  0E DA                     MVI     C,272Q
1219  7E                C1:     MOV     A,M
121A  B8                        CMP     B
121B  3F                        CMC
121C  D2 20 12                  JNC     BAC
121F  B9                        CMP     C
1220  C1                BAC:    POP     B
1221  C9                        RET
1222  C5                ALPHA:  PUSH    B
1223  06 C1                     MVI     B,301Q          ;SET UP ASCII A
1225  0E DB                     MVI     C,333Q
1227  C3 19 12                  JMP     C1
                        ;
                        ; ROUTINE TO CONVERT ASCII NUMERIC CHAR. STRING TO
                        ; EQUIVALENT BINARY NUMBER, RETURNS EQUIVALENT IN
                        ; DE REG. LENGTH OF LINE PASSED IN REG C AND
                        ; RETURNED POINTING TO LAST NUMERIC CHAR. LENGTH
                        ; OF CHAR STRING RETURNED IN REG A.
                        ;
122A  F5                CVB:    PUSH    H
122B  C5                        PUSH    B
122C  CD 71 12                  CALL    LENGTH          ;GET LENGTH OF STRING
122F  F5                        PUSH    PSW
1230  E5                        PUSH    H
1231  FE 00                     CPI     0               ;IS LENGTH=0?
1233  CA 68 12                  JZ      CVB2            ;YEP--GO TO CVB2
1236  21 60 21                  LXI     H,KASE          ;SET UP TEMP VAR KASE
1239  77                        MOV     M,A             ;SAVE LENGTH IN KASE
123A  71                        INR     L               ;SET UP TO MULT. BY
123B  4E                        MOV     C,M             ;DEC 10
123C  21 0A 00                  LXI     H,10
123F  22 62 21                  SHLD    MULT1
1242  21 00 00                  LXI     H,0
1245  22 64 21                  SHLD    MULT2
1248  21 65 21                  LXI     H,MULT2+1
124B  CD E8 0F          CVB1:   CALL    MULT
124E  E3                        XTHL
124F  7E                        MOV     A,M             ;GET NEXT CHAR. AND
1250  DE B0                     SBI     260Q            ;STRIP ASCII FROM IT
1252  82                        ADD     D               ;THE CONVERSION ALGORIT
1253  57                        MOV     D,A
1254  3E 00                     MVI     A,0             ;DE?0
1256  8B                        ADC     E               ;DO WHILE A>0
1257  5F                        MOV     E,A             ;     DE?(HL(M)-260B)+D
1258  23                        INX     H               ;     END:
1259  E3                        XTHL
125A  72                        MOV     M,D
125B  2C                        INR     L
125C  73                        MOV     M,E
125D  E5                        PUSH    H
125E  21 61 21                  LXI     H,LEN
1261  35                        DCR     M
1262  2D                        DCR     L
1263  35                        DCR     M
1264  F1                        POP     H
1265  C2 4B 12                  JNZ     CVB1
1268  F1                CVB2:   POP     H
1269  F1                        POP     PSW             ;THE PROCESS IS COMPLET
126A  C1                        POP     B               ;  RESULT IN DE
126B  21 61 21                  LXI     H,LEN
126E  4F                        MOV     C,M
126F  F1                        POP     H
1270  C9                        RET
                        ;
                        ; ROUTINE TO EVALUATE LENGTH OF ASCII NUMERIC
                        ; CHAR STRING: PASSED ADD OF FIRST CHAR IN HL REG.
                        ; RETURNS LENGTH IN REG A.
                        ;
1271  C5                LENGTH: PUSH    B               ;SAVE REGISTERS
1272  E5                        PUSH    H
1273  06 00                     MVI     B,0
1275  CD 14 12          NLE1:   CALL    NUMB            ;INITIALIZE REG B
1278  D2 84 12                  JNC     NLE2            ;IS HL(M) AN ASCII NUMB
127B  23                        INX     H               ;NO DONE- REG B CONTAIN
127C  04                        INR     B               ;YEP- GET NEXT ONE
127D  0D                        DCR     C
127E  CA 84 12                  JZ      NLE2            ;CHECK FOR END OF LINE
1281  C3 75 12                  JMP     NLE1
1284  78                NLE2:   MOV     A,B             ;LOOP ON PROCESS
1285  E1                        POP     H               ;MOVE LENGTH INTO A
1286  C1                        POP     B
1287  C9                        RET
                        ;
                        ; ROUTINE TO LOCATE SOURCE LINE IN MEM. PASSED BIN VALUE
                        ; OF LINE NUMBER IN DE(LOW,HIGH) REG. RETURNS ADDRESS OF
                        ; SOURCE LINE IN HL REGS.(HIGH,LOW); CY SET=> NOT FOUND.
                        ;
1288  2A 49 21          NSRCH:  LHLD    STLINE          ;GET FIRST LINE
128B  CD A3 12          L2:     CALL    CHK1            ;IS IT -1 (END OF LIST)
128E  D8                        RC                      ;YES- NO SOURCE-RETURN
128F  46                        MOV     B,M             ;MOVE LINE NUMB. INTO B
1290  23                        INX     H
1291  4E                        MOV     C,M
1292  CD F1 0F                  CALL    DCOMP           ;HOW DO THEY COMPARE?
1295  CA A0 12                  JZ      FOUND           ;FOUND
1298  23                        INX     H               ;BUMP HL TO GET POINTER
1299  7E                        MOV     A,M
129A  23                        INX     H
129B  66                        MOV     H,M             ;MOVE TO NEXT LINE AND
129C  6F                        MOV     L,A
129D  C3 8B 12                  JMP     L2
12A0  28                FOUND:  DCX     H               ;MATCH--POINT TO FIRST
12A1  B7                        ORA     A
12A2  C9                        RET
                        ;
                        ; ROUTINE TO COMPARE CONTENTS OF HL TO 177777Q.
                        ; RETURNS CY=1 IF YES: CY=0 IF NO.
                        ;
12A3  C5                CHK1:   PUSH    B               ;SAVE REGISTERS
12A4  E5                        PUSH    H
12A5  06 00                     MVI     B,0             ;MOVE 1 INTO BC
12A7  0E 01                     MVI     C,1
12A9  09                        DAD     B               ;DOUBLE ADD TO SET STAT
12AA  E1                        POP     H
12AB  C1                        POP     B
12AC  C9                        RET
                        ;
                        ; ROUTINE TO PAD OUTPUT BUFFER WITH CONTENTS OF REG A.
                        ; REG B CONTAINS NUMBER OF CHAR TO PAD.
                        ;
12AD  C5                PAD:    PUSH    B               ;SAVE REGISTERS
12AE  D5                        PUSH    D
12AF  E5                        PUSH    H
12B0  21 00 21                  LXI     H,OBUFF         ;GET OBUFF ADDRESS
12B3  4D                        MOV     C,L             ;POINT TO PROPER ENTRY
12B4  6E                        MOV     L,M
12B5  57                        MOV     D,A
12B6  3E 49                     MVI     A,73            ;CHECK FOR LINE LENGTH
12B8  BD                P1:     CMP     L
12B9  C2 C2 12                  JNZ     P2              ;ROOM LEFT--GO FILL
12BC  69                        MOV     L,C             ;OBUFF IS FULL- AUTO PR
12BD  77                        MOV     M,A
12BE  CD D0 12                  CALL    WRIT
12C1  2C                        INR     L               ;RE-INITIALIZE L
12C2  72                P2:     MOV     M,D
12C3  2C                        INR     L
12C4  05                        DCR     B               ;DECREMENT NUMB. CHAR.
12C5  C2 B8 12                  JNZ     P1              ;GO TO P1 IF ANY LEFT
12C8  7A                        MOV     A,D             ;RESTORE PROPER STATUS
12C9  45                        MOV     B,L
12CA  69                        MOV     L,C
12CB  70                        MOV     M,B
12CC  E1                        POP     H
12CD  D1                        POP     D
12CE  C1                        POP     B
12CF  C9                        RET
                        ;
                        ; ROUTINE TO DUMP OUTPUT BUFFER TO TTY.
                        ;
12D0  16 00             WRIT:   MVI     D,0
12D2  F5                WRIT1:  PUSH    PSW
12D3  E5                        PUSH    H
12D4  C5                        PUSH    B
12D5  21 00 21                  LXI     H,OBUFF         ;GET OBUFF ADDR.
12D8  F5                        PUSH    H
12D9  4E                        MOV     C,M             ;GET NUMB. OF CHAR.
12DA  0D                        DCR     C               ;IS OBUFF EMPTY
12DB  CA F0 12                  JZ      W2
12DE  2C                        INR     L
12DF  7E                W1:     MOV     A,M             ;MOVE CHAR.
12E0  F7                        RST     SEND            ;PRINT VIA ODT
12E1  2C                        INR     L               ;MOVE TO NEXT CHAR.
12E2  0D                        DCR     C
12E3  C2 DF 12                  JNZ     W1
12E6  15                        DCR     D
12E7  CA F0 12                  JZ      W2
12EA  3E 8D                     MVI     A,215Q          ;PUT OUT CR, LF.
12EC  F7                        RST     SEND            ;PRINT VIA ODT
12ED  3E 8A                     MVI     A,212Q
12EF  F7                        RST     SEND            ;PRINT VIA ODT
12F0  E1                W2:     POP     H
12F1  36 01                     MVI     M,1
12F3  C1                        POP     B
12F4  F1                        POP     H
12F5  F1                        POP     PSW
12F6  C9                        RET
                        ;
                        ; ROUTINE TO LOCATE COMMANDS, KEY WORDS, OPERATORS,
                        ; AND FUNCTIONS.  HL CONTAINS ADD OF FIRST CHAR.;
                        ; REG C CONTAINS LENGTH OF LINE: RETURNS SYMBOL NUMBER
                        ; IF FOUND IN REG A, 377Q IN A IF NOT FOUND.
                        ; ON ENTRY REG. A CONTAINS TYPE OF ENTRY SOUGHT:
                        ;
                        ;  0 FOR COMMAND
                        ;  1 FOR KEYWORD
                        ;  2 FOR OPERATOR AND DELIMITER
                        ;  3 FOR FUNCTION
                        ;
12F7  D5                SYMSRT: PUSH    D
12F8  C5                        PUSH    B
12F9  E5                        PUSH    H
12FA  E5                        PUSH    H
12FB  21 61 21                  LXI     H,LEN           ;SAVE C IN LEN
12FE  71                        MOV     M,C
12FF  21 4B 13                  LXI     H,KDATA
1302  1E 00                     MVI     E,0
1304  85                       ADD     L
1305  6F                       MOV     L,A
1306  6F                       MOV     L,M
1307  4E                       MOV     C,M
1308  2C                S2:     INR     L               ;LEN IN C AND BUMP L
1309  46                S3:     MOV     B,M             ;FIRST CHAR. IN B
130A  E3                        XTHL                    ;FIRST CHAR OF USER STR
130B  7E                        MOV     A,M             ;AND COMPARE
130C  B8                        CMP     B
130D  C2 23 13                  JNZ     S4              ;NO MATCH-TRY AGAIN
1310  0D                        DCR     C               ;SET UP FOR NEXT CHAR
1311  CA 3F 13                  JZ      S5              ;IF DONE EXIT
1314  F5                        PUSH    PSW
1315  21 61 21                  LXI     H,LEN           ;DECREMENT LINE LENGTH
1318  35                        DCR     M
1319  F1                        POP     PSW
131A  CA 22 13                  JZ      S4A             ;IF 0 THEN MOVE ON
131D  23                        INX     H               ;REPEAT PROCESS
131E  E3                        XTHL
131F  C3 08 13                  JMP     S3
1322  0C                S4A:    INR     C
1323  F1                S4:     POP     H               ;GET ADDR. OF TABLE
1324  79                        MOV     A,C
1325  85                        ADD     L
1326  54                        MOV     D,H
1327  F1                        POP     H
1328  C1                        POP     B
1329  C5                        PUSH    B
132A  E5                        PUSH    H
132B  E5                        PUSH    H
132C  21 61 21                  LXI     H,LEN           ;SET UP PROPER ENTRY
132F  71                        MOV     M,C             ;POINT TO PROPER ENTRY
1330  6F                        MOV     L,A
1331  62                        MOV     H,D
1332  7E                        MOV     A,M
1333  1C                        INR     E
1334  4F                        MOV     C,A
1335  3C                        INR     A
1336  C2 08 13                  JNZ     S3
1339  21 61 21                  LXI     H,LEN
133C  34                        INR     M
133D  1E FF                     MVI     E,377Q          ;E=-1 IF NO MATCH
133F  7B                S5:     MOV     A,E             ;MOVE SYMBOL NUMBER INT
1340  21 61 21                  LXI     H,LEN
1343  5E                        MOV     E,M
1344  1D                        DCR     E
1345  F1                        POP     H
1346  F1                        POP     H
1347  C1                        POP     B
1348  4B                        MOV     C,E             ;MOVE NUMBER OF CHAR. L
1349  D1                        POP     D
134A  C9                        RET
                        ;
                        ;*****************************************************
                        ;THE CODE FROM HERE TO THE NEXT LINE OF *'S MUST BE ON ON
                        ;
                        KDATA:  DB      KDAT1 AND 377Q
                                DB      KDAT2 AND 377Q
                                DB      KDAT3 AND 377Q
                                DB      KDAT4 AND 377Q
1348  4F                KDAT1:  DB      3,322Q,325Q,316Q        ;RUN
1349  9E                        DB      3,323Q,314Q,323Q        ;PLS
134A  C0                        DB      3,314Q,311Q,323Q        ;LIS
134B  C0                        DB      3,321Q,303Q,322Q        ;SCR
134C  C0                        DB      3,320Q,324Q,301Q        ;PTA
134D  FF                        DB      377Q
134E  D3 CC C5 D4       KDAT2:  DB      3,314Q,305Q,324Q        ;LET
1352  D3 D2 C9 CE               DB      3,322Q,322Q,311Q        ;PRI
1356  D3 D2 C5 CD               DB      3,322Q,305Q,315Q        ;REM
135A  D3 D3 D4 CF               DB      3,323Q,324Q,317Q        ;STO
135E  D3 C5 CE C4               DB      3,305Q,316Q,304Q        ;END
1362  D3 C7 CF D4               DB      3,307Q,317Q,324Q        ;GOT
1366  02 C9 C6                  DB      2,311Q,306Q             ;IF
1369  D3 C9 CE D0               DB      3,311Q,316Q,320Q        ;INP
136D  D3 C4 C9 CD               DB      3,304Q,311Q,315Q        ;DIM
1371  C3                        DB      3,'C'+200Q              ;CAL
1372  C1                        DB      'A'+200Q
1373  CC                        DB      'L'+200Q
1374  C7                        DB      'G'+200Q
1375  CF                        DB      'O'+200Q                ;GOSU
1376  D3                        DB      'S'+200Q
1377  D5                        DB      'U'+200Q
1378  D2                        DB      3,'R'+200Q              ;RET
1379  C5                        DB      'E'+200Q
137A  D4                        DB      'T'+200Q
137B  C6                        DB      3,'F'+200Q              ;FOR
```

```
1396  CF                        DB      'O'-'@'+200Q
1397  D2                        DB      'R'-'@'+200Q
1398  04 CE                     DB      4,'N'-'@'+200Q          ;NEXT
139A  C5                        DB      'E'-'@'+200Q
139B  D8                        DB      'X'-'@'+200Q
139C  D4                        DB      'T'-'@'+200Q
139D  FF                        DB      377Q

                        ;DELIMITERS HAVE FOLLOWING VALUES:
                        ;
                        ;       <       0
                        ;       >       1
                        ;       ,       2
                        ;       =       3
                        ;       )       4
                        ;       THEN    5
                        ;       TO      6
                        ;       STEP    7
                        ;       *       8
                        ;       /       9
                        ;
                        ;       +       11
                        ;       -       12
                        ;
139E  01 BC 01 BE       KDAT3:  DB      1,274Q,1,276Q           ;'<','>'
13A2  01 AC 01 BD               DB      1,254Q,1,275Q           ;',','='
13A6  01 A9                     DB      1,251Q                  ;')'
13A8  01 BB                     DB      1,'+'+200Q              ;'+'
13AA  04                        DB      4                       ;THEN
13AB  D4                        DB      'T'-'@'+200Q
13AC  C8                        DB      'H'-'@'+200Q
13AD  C5                        DB      'E'-'@'+200Q
13AE  CE                        DB      'N'-'@'+200Q
13AF  02                        DB      2                       ;TO
13B0  D4                        DB      'T'-'@'+200Q
13B1  CF                        DB      'O'-'@'+200Q
13B2  04                        DB      4                       ;STEP
13B3  D3                        DB      'S'-'@'+200Q
13B4  D4                        DB      'T'-'@'+200Q
13B5  C5                        DB      'E'-'@'+200Q
13B6  D0                        DB      'P'-'@'+200Q
13B7  01 AA                     DB      1,'*'+200Q              ;'*'
13B9  01 AF 01 AB               DB      1,257Q,1,253Q           ;'/','+'
13BD  01 AD                     DB      1,255Q                  ;'-'
13BF  FF                        DB      377Q
13C0  03 C7 C6 D4       KDAT4:  DB      3,307Q,305Q,324Q        ;GET
13C4  03 D0 D5 D4               DB      3,320Q,325Q,324Q        ;PUT
13C8  FF                        DB      377Q

                        ;*****************************************************
                        ; ROUTINE TO INPUT SOURCE LINE FROM TTY. PASSED ADD
                        ; OF FIRST CHAR IN HL. RETURNS LENGTH OF LINE IN REG A
13C9  F5                TTYIN:  PUSH    H
13CA  06 00                     MVI     B,0
13CC  CD F7 0F          TIN1:   CALL    CHAR2
13CF  FE 99                     CPI     231Q                    ;CNTRL Y?
13D1  CA D1 14                  JZ      TIN5
13D4  FE FF                     CPI     377Q                    ;RUBOUT?
13D6  CA F6 13                  JZ      TIN2
13D9  FE DF                     CPI     337Q                    ;BACK ARROW (RUBOUT)?
13DB  CA F9 13                  JZ      TIN2+3
13DE  FE 8A                     CPI     212Q                    ;LF?
13E0  CA CC 13                  JZ      TIN1
13E3  FE 8D                     CPI     215Q                    ;CR
13E5  CA 08 14                  JZ      TIN4
13E8  FE 8C                     CPI     214Q                    ;FORM FEED?
13EA  CA CC 13                  JZ      TIN1                    ;IGNORE
13ED  77                        MOV     M,A
13EE  23                        INX     H
13EF  04                        INR     B
13F0  CD 14 15                  CALL    MEMFUL
13F3  C3 CC 13                  JMP     TIN1
13F6  3E DF             TIN2:   MVI     A,337Q
13F8  E7                        RST     SEND                    ;PRINT VIA ODT
13F9  2B                        DCX     H
13FA  05                        DCR     B
13FB  F2 CC 13                  JP      TIN1
13FE  E1                        POP     H
13FF  AF                        XRA     A                       ;ZERO A
1400  C9                        RET
1401  3E 0C              TIN5:   MVI     A,334Q
1403  F7                        RST     SEND                    ;PRINT VIA ODT
1404  3E 00             TIN5A:  MVI     A,0
1406  F1                        POP     H
1407  C9                        RET
1408  3E 8A             TIN4:   MVI     A,212Q
140A  F7                        RST     SEND                    ;PRINT VIA ODT
140B  0E 00             TIN4A:  MVI     C,0
140D  E1                        POP     H
140E  78                        MOV     A,B
140F  B9                        CMP     C
1410  C8                        RZ

                        ;ROUTINE TO REMOVE BLANKS FROM SOURCE UNLESS ENCLOSED IN
1411  D5                        PUSH    D                       ;SAVE REG'S
1412  E5                        PUSH    H
1413  E5                        PUSH    H
1414  1E A2                     MVI     E,'"'+200Q              ;INIT E FOR COMPARES
1416  16 00                     MVI     D,0                     ;D=1=>WITHIN QUOTES, LE
1418  AF                PK1:    XRA     A                       ;CLEAR A
1419  BA                        CMP     D                       ;CHECK INPUT MODE
141A  7E                        MOV     A,M                     ;GET CHAR
141B  C2 2A 14                  JNZ     QSTR.                   ;WITHIN QUOTE STRING
141E  BB                        CMP     E                       ;IS IT 1ST "?
141F  C2 26 14                  JNZ     $+7                     ;NO - PROCEED
1422  14                        INR     D                       ;YES, SET FLAG
1423  C3 30 14                  JMP     QSTR1                   ;CONTINUE
1426  FE A0               QSTR:   CPI     240Q                    ;IS IT A SPACE?
1428  CA 35 14                  JZ      PK2                     ;YES - LEAVE OUT
142B  BB                        CMP     E                       ;2ND "?
142C  C2 30 14                  JNZ     $+4                     ;NO - CONTINUE
142F  15                        DCR     D                       ;RESET FLAG
1430  E3                QSTR1:  XTHL                            ;GET SOURCE ADDRES
1431  77                        MOV     M,A                     ;SAVE
1432  23                        INX     H                       ;BUMP PNTR.
1433  E3                        XTHL                            ;GET SOURCE ADD.
1434  0C                        INR     C                       ;BUMP CHAR. CNT
1435  23                PK2:    INX     H                       ;BUMP PNTR.
1436  05                        DCR     B                       ;DCR INPUT LINE CHAR CN
1437  C2 18 14                  JNZ     PK1                     ;MORE - GO AGAIN
143A  79                        MOV     A,C                     ;CHAR CNT TO A
143B  F1                        POP     H                       ;RESTORE REG'S, RETURN
143C  F1                        POP     H
143D  D1                        POP     D
143E  C9                        RET

                        ;ROUTINES TO  PAD MESSAGES TO  OUTPUT BUFFER.
                        ; FORM12 PADS 'UNDERFLOW'
                        ; FORM11 PADS 'OVERFLOW'
                        ; FORM10 PADS 'ZERODIVIDE'
                        ; FORM9 PADS 'INPUT ERROR, TRY AGAIN'
                        ; FORM8 PADS 'MEMORY FULL'
                        ; FORM7 PADS 'WHAT?'
                        ; FORM4 PADS 'IN LINE'
                        ; FORM3 PADS 'ERROR'
                        ; FORM2 PADS 'TURN ON PUNCH'
                        ; FORM1 PADS 'READY'
                        ; FORM5 PADS SOURCE LINE, PASSED ADDRESS OF
                        ; LENGTH OF LINE IN HL REGS.
                        ; FORM6 PADS CHAR STRING, PASSED ADD OF FIRST CHAR IN
                        ; HL, LENGTH OF STRING IN REG C
143F  2C                FORM12: INR     L                       ;THE ENTRY POINT INCREM
1440  2C                FORM11: INR     L                       ;PROPER DATA POINT
1441  2C                FORM10: INR     L
1442  2C                FORM9:  INR     L
1443  2C                FORM8:  INR     L
1444  2C                FORM7:  INR     L
1445  2C                FORM4:  INR     L
1446  2C                FORM3:  INR     L
1447  2C                FORM2:  INR     L
1448  6F                FORM1:  MOV     L,M                     ;POINT TO PROPER BUFFER
1449  4E                FORM5:  MOV     C,M                     ;MOVE LENGTH INTO C
144A  79                        MOV     A,C                     ;AND STORE IN A
144B  FE 00                     CPI     0                       ;IS IT 0?
144D  C8                        RZ
144E  23                F1:     INX     H                       ;INCREMENT TO GET FIRST
144F  7E                FORM6:  MOV     A,M                     ;THE PAD LOOP
1450  06 01                     MVI     B,1
1452  CD A0 12                  CALL    PAD
1455  0D                        DCR     C
1456  C2 4E 14                  JNZ     F1
1459  C9                        RET

                        ;*****************************************************
                        ; THE CODE FROM HERE TO THE NEXT LINE OF **S MUST BE ON O
145A  64                ODATA:  DB      ODAT1 AND 377Q
145B  6A                        DB      ODAT2 AND 377Q
145C  78                        DB      ODAT3 AND 377Q
145D  81                        DB      ODAT4 AND 377Q
145E  8B                        DB      ODAT5 AND 377Q
145F  91                        DB      ODAT6 AND 377Q
1460  A0                        DB      ODAT7 AND 377Q
1461  B7                        DB      ODAT8 AND 377Q
1462  C2                        DB      ODAT9 AND 377Q
1463  CB                        DB      ODA10 AND 377Q
1464  05 52 45 41     ODAT1:  DB      5,'READY'
1468  44 59
146A  0D 54 55 52       ODAT2:  DB      13,'TURN ON PUNCH'
146E  4E 20 4F 4E
1472  20 50 55 4E
1476  43 48
1478  0B 8D 8A 45     ODAT3:  DB      8,215Q,212Q,'ERROR '
147C  52 52 4F 52
1480  20
1481  09 20 49 4E     ODAT4:  DB      9,' IN LINE '
1485  20 4C 49 4E
1489  45 20
148B  05 57 48 41       ODAT5:  DB      5,'WHAT?'
148F  54 3F
1491  0E 4D 45 40     ODAT6:  DB      14,'MEMORY FULL',215Q,212Q,'?'
1495  4F 52 59 20
1499  46 55 4C
149D  8D 8A 3F
14A0  16 49 4E 50       ODAT7:  DB      22,'INPUT ERROR, TRY AGAIN'
14A4  55 54 20 45
14A8  52 52 4F 52
14AC  2C 20 54 52
14B0  59 20 41 47
14B4  41 49 4E
14B7  0A 49 4E 44       ODAT8:  DB      10,'INDEFINITE'
14BB  45 46 49 4E
14BF  49 54 45
14C2  08 4F 56 45       ODAT9:  DB      8,'OVERFLOW'
14C6  52 46 4C 4F
14CA  57
14CB  09 55 4E 44       ODA10:  DB      9,'UNDERFLOW'
14CF  45 52 46 4C
14D3  4F 57

                        ;*****************************************************
                        ; ROUTINE TO INPUT SOURCE LINE FROM HSR. PASSED ADD
                        ; OF FIRST CHAR IN HL. RETURNS LENGTH OF LINE IN REG A
14D5  E5                HSRIN:  PUSH    H                       ;SAVE ADDR.
14D6  06 00                     MVI     B,0                     ;INIT. NUMB. OF CHAR. R
14D8  C3 DE 14                  JMP     PIN1A
14DB  CD DB 15          PIN1:   CALL    CHAR5                   ;GET A CHAR.
14DE  FE 99             PIN1A:  CPI     231Q                    ;CNTRL Y?
14E0  CA 04 14                  JZ      TIN5A
14E3  FE FF                     CPI     377Q                    ;DEL?
14E5  CA 00 15                  JZ      PIN3

14E8  FE 0F                     CPI     337Q                    ;CNTR A?
14EA  CA 03 15                  JZ      PIN3
14ED  FE 8A                     CPI     212Q                    ;CR?
14EF  CA 0A 14                  JZ      TIN4A
14F2  FE 8D                     CPI     215Q                    ;CR?
14F4  CA DB 14                  JZ      PIN1
14F7  77                        MOV     M,A                     ;GOOD CHAR--STORE IT AW
14F8  23                        INX     H                       ;MOVE TO NEXT SLOT
14F9  04                        INR     B
14FA  CD 14 15                  CALL    MEMFUL                  ;IS MEM. FULL?
14FD  C3 DB 14                  JMP     PIN1                    ;LOOP ON PROCESS
1500  2B                PIN3:   DCX     H                       ;DELETE THE LAST CHAR.
1501  05                        DCR     B                       ;MUST INSURE THE LINE I
1502  F2 DB 14                  JP      PIN1
1505  F1                        POP     H
1506  AF                        XRA     A                       ;ZERO A
1507  C9                        RET

                        ; ROUTINE TO INPUT CHAR FROM HSR
1508  C5                CHAR5:  PUSH    B
1509  06 04                     MVI     B,4                     ;THE INPUT REQUEST
150B  D3 05                     OUT     5
150D  CD F4 0F                  CALL    MCHK                    ;GO GET CHAR.
1510  DB 05                     IN      5                       ;THE TERMINATE REQUEST
1512  C1                        POP     B
1513  C9                        RET

                        ; ROUTINE TO INSURE SOURCE DOES NOT OVERFLOW MEM SPACE
                        ; COMPARES CURRENT MEM ADDRESS TO SP.
1514  C5                MEMFUL: PUSH    B                       ;SAVE REG.B,D,H
1515  D5                        PUSH    D
1516  E5                        PUSH    H
1517  3E 32                     MVI     A,50                    ;MOVE DEC 50 INTO B TO
1519  85                        ADD     L                       ;INSURE MARGINE OF STAC
151A  47                        MOV     B,A
151B  3E 00                     MVI     A,0
151D  8C                        ADC     H
151E  4F                        MOV     C,A                     ;ESTABLISH MEM+50 IN BC
151F  21 00 00                  LXI     H,0
1522  39                        DAD     SP                      ;MOVE SP INTO HL
1523  55                        MOV     D,L
1524  5C                        MOV     E,H                     ;SET UP FOR DCOMP
1525  CD F1 0F                  CALL    DCOMP
1528  E1                        POP     H
1529  D1                        POP     D
152A  C1                        POP     B
152B  D0                        RNC                             ;CY=0 THEN OK TO CONTIN
152C  21 5A 14                  LXI     H,ODATA
152F  CD 43 14                  CALL    FORM8                   ;MEMORY FULL--PAD
1532  CD D0 12                  CALL    WRIT                    ;MESSAGE AND WRITE
1535  CD F7 0F                  CALL    CHAR2                   ;GET USER RESPONCE
1538  CD A0 12                  CALL    PAD                     ;MUST BE 1,2 OR 3 FOR C
153B  CD D0 12                  CALL    WRIT
153E  DE B0                     SBI     260Q
1540  FE 04                     CPI     4
1542  CC 9E 11                  CZ      WHAT                    ;IMPROPER RESPONCE
1545  31 F5 27                  LXI     SP,MEMEND-10            ;RESET SP
1548  0E 01                     MVI     C,1
154A  C3 47 10                  JMP     M4A                     ;LOOP ON CONTROL SECTIO

                        ; ROUTINE TO EVALUATE BOUNDS FOR LIST AND PLIST
                        ; COMMANDS. RETURNS PLINE AS FIRST LINE, KLINE
                        ; AS LAST LINE TO BE LISTED.
154D  2A 4B 21          BOUND:  LHLD    NLINE                   ;GET ADDR. OF NEW LINE
1550  3E 09                     MVI     A,9
1552  85                        ADD     L                       ;INCREMENT 9 TO GET TO
1553  6F                        MOV     L,A
1554  3E 00                     MVI     A,0
1556  8C                        ADC     H                       ;THE ADDR. IS NOW THERE
1557  67                        MOV     H,A
1558  E5                        PUSH    H
1559  CD 14 12                  CALL    NUMB                    ;SEE IF THERE IS A NUMB
155C  D4 9E 11                  CNC     WHAT                    ;IT IS NOT
155F  CD 2A 12                  CALL    CVB                     ;CONVERT NUMBER TO BINA
1562  F5                        PUSH    PSW
1563  C5                        PUSH    B
1564  CD A3 15                  CALL    BND2                    ;GO FIND FIRST LINE
1567  C1                        POP     B
1568  2B                        DCX     H
1569  22 59 21                  SHLD    PLINE                   ;STORE IT WAY AS FIRST
156C  F1                BND1:   POP     PSW
156D  F1                        POP     H
156E  3C                        INR     A
156F  85                        ADD     L                       ;SET UP TO GET UPPER BO
1570  6F                        MOV     L,A
1571  3E 00                     MVI     A,0
1573  8C                        ADC     H
1574  67                        MOV     H,A                     ;UPPER BOUND ADDR IN HL
1575  3E 00                     MVI     A,0
1577  B9                        CMP     C                       ;IS THAT END OF LINE?
1578  C8                        RZ
1579  0D                        DCR     C
157A  CD 14 12                  CALL    NUMB                    ;IS NEXT CHAR NUMB?
157D  D4 9E 11                  CNC     WHAT                    ;NOT NUMBER --WHZTS UP
1580  D5                        PUSH    D
1581  CD 2A 12                  CALL    CVB                     ;CONVERT TO BINARY
1584  D5                        PUSH    D
1585  C5                        PUSH    B
1586  CD A3 15                  CALL    BND2                    ;GO FIND SECOND BOUND
1589  C1                        POP     B
158A  23                        INX     H
158B  56                        MOV     D,M
158C  23                        INX     H
158D  5E                        MOV     E,M                     ;SETUP TO STORE IT IN K
158E  EB                        XCHG
158F  22 52 21                  SHLD    KLINE
1592  D1                        POP     D
1593  E1                        POP     H
1594  79                        MOV     A,C
1595  FE 00                     CPI     0                       ;ARE WE AT END OF LINE
1597  C2 9E 11                  JNZ     WHAT
159A  44                        MOV     B,H
159B  4D                        MOV     C,L
159C  CD F1 0F                  CALL    DCOMP
159F  D0                        RNC
15A0  C3 9E 11                  JMP     WHAT
15A3  2A 49 21          BND2:   LHLD    STLINE                  ;SET UP TO SEARCH FROM
15A6  46                BND3:   MOV     B,M
15A7  23                        INX     H
15A8  4E                        MOV     C,M
15A9  CD F1 0F                  CALL    DCOMP                   ;COMPARE BINARY OF REQU
15AC  D8                        RC                              ;TO CURRENT LINE
15AD  C8                        RZ
15AE  E5                        PUSH    H
15AF  23                        INX     H
15B0  7E                        MOV     A,M
15B1  23                        INX     H
15B2  66                        MOV     H,M
15B3  6F                        MOV     L,A                     ;MOVE TO NEXT LINE
15B4  CD A3 12                  CALL    CHK1                    ;IS THAT ALL?
15B7  C1                        POP     B
15B8  D2 A6 15                  JNC     BND3                    ;NOPE--LOOP ON PROCESS
15BB  C5                        PUSH    B
15BC  E1                        POP     H
15BD  C9                        RET
```

```
                        ; ROUTINE TO OUTPUT ERROR MSG. TO USER.
                        ; REG A CONTAINS BCD ERROR NUMBER, HL
                        ; LOADED WITH VALUE OF KLINE.
                        ;
158E  21 13 10  ERROR:  LXI   H,M1A             ;RETURN ADDRESS
158I  E5        ERR0:   PUSH  H                 ;PUT ON STACK
15C2  21 5A 14          LXI   H,0DATA           ;OUTPUT BUFFER DATA TAB
15C5  E5                PUSH  H
15C6  57                MOV   D,A               ;SAVE ERROR NUMB. IN D
15C7  CD 46 14          CALL  FORM3             ;PAD 'ERROR '
15CA  06 01             MVI   B,1               ;INIT FOR PADS
15CC  48                MOV   C,B               ;INIT AS CNTR.
15CD  7A                MOV   A,D               ;GET ERROR NUMB.
15CE  07                RLC                     ;ROTATE HIGH 4 BITS TO
15CF  07                RLC
15D0  07                RLC
15D1  07                RLC
15D2  E6 0F     ERRR1:  ANI   17Q               ;MASK
15D4  C6 B0             ADI   260Q              ;CONVERT TO ASCII
15D6  CD AU 12          CALL  PAD               ;PAD IT
15D9  7A                MOV   A,D               ;GET ERROR NUMB.
15DA  0D                DCR   C                 ;ANOTHER PASS?
15DB  F2 D2 15          JP    ERRR1             ;YES
15DE  L1                POP   H                 ;NO-CONTINUE
15DF  CD 45 14  ERLN:   CALL  FORM4             ;PAD 'IN LINE'
15E2  2A 52 21          LHLD  KLINE
15E5  23                INX   H
15E6  23                INX   H
15E7  23                INX   H
15E8  23                INX   H
15E9  4E                MOV   C,M
15EA  23                INX   H
15EB  CD 71 12          CALL  LENGTH
15EE  4F                MOV   C,A
15EF  CD 4F 14          CALL  FORM6             ;PAD LINE #
15F2  CD DO 12          CALL  WRIT              ;WRITE MESSAGE
15F5  C9                RET
                        ;
                        ; THIS ROUTINE INCREMENTS H AND L AND
                        ; DECR. C(CHARS IN LINE) SHOULD C RESULT
                        ; IN 0 THEN THE ERROR CORRES. TO ENTRY PNT.
                        ; IS GIVEN
                        ;
15F6  3E 07     ICP7:   MVI   A,7
15F8  C3 07 16          JMP   INCPT
15FB  3E 08     ICP8:   MVI   A,8
15FD  C3 07 16          JMP   INCPT
1600  3E 04     ICP4:   MVI   A,4
1602  C3 07 16          JMP   INCPT
1605  3E 02     ICP2:   MVI   A,2
1607  23        INCPT:  INX   H
1608  0D                DCR   C
1609  C0                RNZ
160A  C3 BE 15          JMP   ERROR
                        ;
                        ; FSYM FINDS SYMBOLS IN TABLE
                        ; B,C CONTAIN SYMBOL
                        ; RET WITH B,C,D,E SAME
                        ; H AND L PNT TO VALUE (1ST BYTE)
                        ; CY=1   => FOUND
                        ; CY=0   AND A SCALAR VAR. => INSERTED
                        ;     AND SET TO 0
                        ; CY=0   AND AN ARRAY  => NO ACTION,
                        ;    H AND L PNT TO LAST ENTRY IN SYMBOL TABLE
                        ;
160D  D5        FSYM:   PUSH  D
160E  AF                XRA   A
160F  HQ                ORA   B                 ;SET CARRY IF NOT
1610  CA 14 16          JZ    AR                ;AN ARRAY AND SAVE
1613  3F                CMC
1614  F5        AR:     PUSH  PSW
1615  2A 59 21          LHLD  NXTSP             ;GET NEXT AVAILABLE
1618  C5                PUSH  B                 ;SPACE PNTR.
1619  44                MOV   B,H
161A  4D                MOV   C,L               ;CHECK TO SEE
161B  2A 4B 21          LHLD  STSPAC            ;IF SYMBOL TABLE
161E  54                MOV   D,H               ;EMPTY
161F  5D                MOV   E,L
1620  CD F1 0F          CALL  DCOMP             ;DOUBLE BYTE COMPARE
1623  C1                POP   B                 ;GET VAR. BACK
1624  CA 44 16          JZ    NOSYM
1627  CD A3 12  LUKUN:   CALL  CHK1              ;CHECK FOR END
162A  DA 46 16          JC    NOENT
162D  54                MOV   D,H               ;SAVE OLD PNTR
162E  5D                MOV   E,L
162F  78                MOV   A,B
1630  BE        CMP     M                       ;DO VARIABLES MATCH
1631  C2 3B 16          JNZ   NOMAT
1634  23                INX   H
1635  79                MOV   A,C
1636  BE                CMP   M
1637  CA 71 16          JZ    ENTRY
163A  2B                DCX   H
163B  23        NOMAT:  INX   H                 ;NO MATCH GET NEW PNT.
163C  23                INX   H
163D  7E                MOV   A,M
163E  23                INX   H
163F  66                MOV   H,M
1640  6F                MOV   L,A
1641  C3 27 16          JMP   LUKUN
                        ; ARRIVE HERE IF SYMBOL TABLE IS EMPTY
1644  1B       NOSYM:   DCX   D                 ; =STSPAC-2 SO STPNT WO
1645  1B                DCX   D
                        ; ARRIVE HERE WHEN NO ENTRY FOUND
1646  2A 59 21  NOENT:  LHLD  NXTSP             ;ADD. OF FREE MEMORY
1649  EB                XCHG                    ;TO DE, HL HAVE LAST SY
164A  F1                POP   PSW               ;ARRAY?
164B  D2 76 16          JNC   FBAC              ;YES, RETURN
164E  CD 80 11          CALL  STPNT             ;UPDATE PNTR
1651  EB                XCHG                    ;NXTSP TO HL
1652  70                MOV   M,B               ;STORE VAR.
1653  23                INX   H
1654  71                MOV   M,C
1655  23                INX   H
1656  E5                PUSH  H
1657  23                INX   H                 ;STORE NXTSP+8 IN NXTSP
1658  23                INX   H
1659  23                INX   H
165A  23                INX   H
165B  23                INX   H
165C  23                INX   H
165D  22 59 21          SHLD  NXTSP
1660  CD 14 15          CALL  MEMFUL            ;MEMORY FULL?
1663  F1                POP   H                 ;SET FWD PNT. TO -1
1664  36 FF             MVI   M,377Q
1666  23                INX   H
1667  36 FF             MVI   M,377Q
1669  23                INX   H                 ;INIT TO FLT. PNT. 0
166A  CD CD 0F          CALL  ZROL
166D  B7                ORA   A                 ;CLEAR CY
166E  C3 76 16          JMP   FBAC              ;RESET CARRY AND RETURN
1671  F1        ENTRY:   POP   PSW               ;VAR FOUND
1672  23                INX   H                 ;MOVE PNT. TO FIRST BYT
1673  23                INX   H                 ;OF FLT. PNT. NO.
1674  23                INX   H
1675  37                STC                     ;SET CY AND RET.
1676  01        FBAC:    POP   D                 ;RESTORE D
1677  C9                RET
                        ;
                        ; RUN - THE INTERP.
                        ;
                        ; INIT. NXTSP
                        ;
1678  2A 4B 21  RUN:     LHLD  STSPAC
167B  22 59 21          SHLD  NXTSP
167E  21 AA 21          LXI   H,BOTNS           ;INIT SP FOR NESTING ST
1681  22 94 21          SHLD  NEST
1684  21 13 10          LXI   H,M1A             ;PRECAUTION, IN CASE RE
1687  E5                PUSH  H                 ;EXECUTED BEFORE A GOSU
1688  E5                PUSH  H
1689  2A 49 21          LHLD  STLINE            ;START OF SOURCE
168C  CD 08 12  ILOOP:   CALL  QUIT              ;CHECK FOR INTERRUPTION
168F  CD A3 12           CALL  CHK1              ;HL=-1 => NO MORE SOURC
1692  D2 9A 16          JNC   SORCE
1695  3E 01             MVI   A,1
1697  C3 BE 15          JMP   ERROR             ;ERROR 1, NO END STMT.
169A  22 52 21  SORCE:   SHLD  LPNT
169D  E5                PUSH  H
169E  21 52 21           LXI   H,LPNT            ;DEFINE VALUES OF
16A1  CD EE 0F           CALL  PTVAL             ;KFPNT,KLEN
16A4  3A 58 21           LDA   KLEN              ;CHAR'S IN LINE TO C
16A7  4F                 MOV   C,A
16A8  0C                 INR   C
16A9  E1                 POP   H                 ;MOVE PNTR. TO 1ST CHAR
16AA  23                 INX   H                 ;IN SOURCE REC.
16AB  23                 INX   H
16AC  23                 INX   H
16AD  23                 INX   H
16AE  CD 05 16  L1:      CALL  ICP2              ;INCR. H,L DCR C
16B1  CD 22 12           CALL  ALPHA             ;FIND FIRST LETTER
16B4  D2 AE 16           JNC   L1
16B7  AF                 XRA   A
16B8  3C                 INR   A                 ;LETTER FOUND
16B9  CD F7 12           CALL  SYMSRT            ;DETERMINE KEYWORD
16BC  FE FF              CPI   377Q
16BE  C2 C6 16           JNZ   GKEY
16C1  3E 02              MVI   A,2               ;BAD KEYWORD
16C3  C3 BE 15           JMP   ERROR
16C6  22 5B 21  GKEY:    SHLD  CPNT
16C9  21 D6 16           LXI   H,JTBL            ;LOAD JUMP TABLE PNTR.
16CC  87                 ADD   A                 ;DOUBLE A
16CD  5F                 MOV   E,A
16CE  16 00              MVI   D,0
16D0  19                 DAD   D                 ;PNT. TO PROPER PROC.
16D1  7E                 MOV   A,M               ;ADD. IN JUMP TABLE
16D2  23                 INX   H                 ;GET PROC. ADD.
16D3  66                 MOV   H,M
16D4  6F                 MOV   L,A
16D5  E9                 PCHL                    ;INDIRECT JUMP TO PROC.
16D6  0D 1B      JTBL:   DW    LET               ;JMP TABLE
16D8  95 1A              DW    PRI
16DA  8C 1B              DW    IEND              ;REM STMT. - NO ACTION
16DC  13 10              DW    M1A               ;STOP STMT.-RETURN TO E
16DE  F2 1C              DW    ENDD
16E0  00 17              DW    GOTO
16E2  7D 1C              DW    IFRT
16E4  4C 18              DW    INPUT
16E6  20 17              DW    DIM
16E8  5A 1D              DW    CALLP
16EA  00 1D              DW    GOSUB
16EC  E5 1D              DW    RETRN
16EE  E7 1D              DW    FOR
16F0  D9 1E              DW    NEXT
                        ;
                        ; END PROCESSOR
                        ;
16F2  2A 56 21  ENDD:    LHLD  KFPNT             ;CHECK TO SEE IF MORE
16F5  CD A3 12           CALL  CHK1              ;SOURCE AFTER END
16F8  DA 13 10           JC    M1A
16FB  3E 03              MVI   A,3               ;MORE SOURCE ERROR 3
16FD  C3 BE 15           JMP   ERROR
                        ;
                        ; GO TO PROCESSOR
                        ;
1700  2A 5B 21  GOTC:    LHLD  CPNT              ;GOTO STMT. PROC.
1703  23       GSENT:   INX   H                 ;INCREMENT PAST KEYWORD
1704  23                INX   H
1705  23                INX   H
1706  CD 00 16          CALL  ICP4              ;POSSIBLE ERROR 4
1709  CD 2A 12  GTRA:   CALL  CVB               ;GET DESTINATION
170C  B7                ORA   A                 ;MAKE SURE IT WAS OK
170D  C2 15 17          JNZ   OKN
1710  3E 04             MVI   A,4
1712  C3 BE 15          JMP   ERROR
1715  CD B8 12  OKN:    CALL  NSRCH             ;GET NEXT LPNT
1718  D2 8C 16          JNC   ILOOP             ;MAKE SURE IT EXISTED
171B  3E 05             MVI   A,5
171D  C3 BE 15          JMP   ERROR             ;NON-EXISTENT
                        ;
                        ; DIMENSION PROCESSOR
                        ;
1720  2A 5B 21  DIM:     LHLD  CPNT              ;DIM STMT. PROC.
1723  23                INX   H                 ;PNT TO FIRST VAR.
1724  23                INX   H
1725  23                INX   H
1726  CD 22 12  DLOOP:   CALL  ALPHA             ;CHECK IF IT IS A VAR.
1729  DA 31 17          JC    OKLET
172C  3E 06             MVI   A,6               ;ERROR 6
172E  C3 BE 15          JMP   ERROR
1731  46        OKLFT:   MOV   B,M
1732  CD F6 15          CALL  ICP7              ;INCR.CPNT
1735  3E A6             MVI   A,250Q            ;CHECK FOR (
1737  BE                CMP   M
1738  C2 2C 17          JNZ   ER6
173B  CD F6 15          CALL  ICP7              ;INCR. CPNT
173E  CD 2A 12          CALL  CVB               ;CONV TO BIN NO.
1741  85                ADD   L                 ;UPDATE CPNT
1742  6F                MOV   L,A               ;ED CONTAIN ARRAY LEN.
1743  3E 00             MVI   A,0
1745  8C                ADC   H
1746  67                MOV   H,A               ;C CONT. NO. CHARS LEFT
1747  3E A9             MVI   A,251Q            ;IN LINE
1749  BE                CMP   M                 ;CHECK FOR )
174A  C2 2C 17          JNZ   ER6
174D  E5                PUSH  H
174E  C5                PUSH  B                 ;SAVE B,C,H,L
174F  48                MOV   C,B               ;SET UP FOR CALL TO FSY
1750  06 00             MVI   B,0
1752  CD 00 16          CALL  FSYM
1755  D2 5F 17          JNC   NDOU
1758  C1                POP   B
1759  E1                POP   H
175A  3E 11             MVI   A,11H             ;ERROR 11
175C  C3 BE 15          JMP   ERROR             ;DUPLICATE ARRAY DEF.
175F  D5        NDOU:    PUSH  D                 ;SAVE DIM. LENGTH
1760  EB                XCHG                    ;ADD. OF LAST SYM. TAB.
1761  2A 59 21          LHLD  NXTSP             ;GET ADD. OF AVAILABLE
1764  EB                XCHG                    ;SET UP FOR CALL
1765  CD BB 11          CALL  STPNT             ;STORE NEW PNTR
1768  EB                XCHG                    ;NXTSP TO HL
1769  D1                POP   D                 ;RESTORE D
176A  36 00             MVI   M,0
176C  23                INX   H                 ;INSERT VAR IN SYMB. TA
176D  71                MOV   M,C
176E  23                INX   H
176F  36 FF             MVI   M,377Q            ;FPNT TO -1
1771  23                INX   H
1772  36 FF             MVI   M,377Q
1774  23                INX   H                 ;PNTS TO FIRST DATA .
1775  7A                MOV   A,D               ;GET ONE'S COMPLEMENT O
1776  2F                CMA                     ;NUMBER OF ELEMENTS
1777  4F                MOV   C,A               ;IN ARRAY TO B,C
1778  7B                MOV   A,E
1779  2F                CMA
177A  47                MOV   B,A
177B  CD CD 0F  CONT:    CALL  ZROL              ;ZERO OUT ELEMTS.
177E  23                INX   H                 ; OF ARRAY
177F  23                INX   H
1780  23                INX   H
1781  23                INX   H
1782  03                INX   B
1783  F5                PUSH  H
1784  CD 14 15          CALL  MEMFUL            ;MEMORY FULL?
1787  60                MOV   H,B
1788  69                MOV   L,C
1789  CD A3 12          CALL  CHK1
178C  E1                POP   H
178D  D2 76 17          JNC   CONT
1790  22 59 21          SHLD  NXTSP             ;NEW VALUE OF NXTSP.
1793  C1                POP   B                 ;RESTORE REG'S
1794  E1                POP   H
1795  23                INX   H
1796  0D                DCR   C                 ;MORE ELEMTS IN LINE?
1797  CA 8C 1B          JZ    IEND
179A  0D                DCR   C
179B  CA 2C 17          JZ    ER6
179E  3E AC             MVI   A,254Q            ;NEXT ELEMENT A ,
17A0  BE                CMP   M
17A1  23                INX   H
17A2  CA 26 17          JZ    DLOOP
17A5  C3 2C 17          JMP   ER6
                        ;
                        ; ROUTINE TO COPY CONTENTS PNTED TO
                        ; BY DE TO LOCATION H,L
                        ;
17A8  F5        COPDH:   PUSH  PSW               ;SAVE REGISTERS
17A9  C5                 PUSH  B
17AA  D5                 PUSH  D
17AB  E5                 PUSH  H
17AC  06 04              MVI   B,4               ;COUNT
17AE  1A        COPD1:   LDAX  D                 ;GET FROM SOURCE
17AF  77                 MOV   M,A               ;PUT TO DESTINATION
17B0  13                 INX   D                 ;BUMP PNTRS, CNT
17B1  23                 INX   H
17B2  05                 DCR   B
17B3  C2 AE 17           JNZ   COPD1
17B6  E1                 POP   H                 ;RESTORE REGISTERS
17B7  D1                 POP   D
17B8  C1                 POP   B
17B9  F1                 POP   PSW
17BA  C9                 RET
                        ;
                        ; OUTR PADS OUTPUT FROM CONV INTO
                        ; OUTPUT BUFFER USING ROUTINE PAD
                        ; ALL REG'S MAINTAINED
                        ;
17BB  C5        OUTR:    PUSH  B                 ;SAVE REG B
17BC  06 01              MVI   B,1               ;PAD ONCE
17BE  CD AD 12           CALL  PAD               ;DO IT
17C1  C1                 POP   B                 ;RESTORE B AND RET.
17C2  C9                 RET
                        ;
                        ; VALUE RETURNS IN D(H),E(L) PNTR.
                        ; TO THE VALUE OF A TOKEN
                        ; C,H,L ARE UPDATED
                        ; A,B ARE DESTROYED
                        ;
17C3  CD 6D 18  VALUE:   CALL  VAR               ;IS IT A VARIABLE?
17C6  D8                 RC                      ;YES - ALL DONE
17C7  3E 03              MVI   A,3               ;NO CHEK IF A FUNC.
17C9  CD F7 12           CALL  SYMSRT
17CC  FE FF              CPI   377Q
17CE  CA 3B 18           JZ    KONT              ;NOT A FUNCTION -
17D1  FE 01              CPI   1                 ;WAS IT PUT(--)?
17D3  C2 D9 17           JNZ   GET               ;NO - OK
17D6  C3 7d 1C           JMP   ER10              ;ILLEGAL USE OF FUNCTIO
17D9  23         GET:    INX   H                 ;OK, IT'S GET(--)
17DA  23                 INX   H                 ;UPDATE H,L
17DB  23                 INX   H
17DC  79                 MOV   A,C               ;CHECK FOR PREMATURE EO
17DD  B7                 ORA   A
17DE  CA 22 1A           JZ    ER8
17E1  3E A8              MVI   A,250Q            ;CHEK FOR (
17E3  BE                 CMP   M
17E4  C2 22 1A           JNZ   ER8
17E7  CD F6 15           CALL  ICP8              ;BUMP PNTR'S
17EA  CD C7 19           CALL  EVAL              ;GET PORT #
```

```
17E0  E5              PUSH  H               ;SAVE REG H,L
17EE  21 7C 21        LXI   H,FREG1
                      CALL  COPDH           ;COPY IT
17F1  CD A8 17        XCHG
17F4  EB              POP   H               ;RESTORE H,L
17F5  E1              CALL  FIX             ;FIX IT
17F6  CD 15 19        INX   D
17F9  13              INX   D               ;GET LOWEST BYTE TO
17FA  13              LDAX  D               ;REG D
17FB  13              MOV   D,A
17FC  1A              MOV   A,C             ;EOL?
17FD  57              ORA   A
17FE  79              JZ    ER8             ;CHECK FOR )
17FF  B7              MVI   A,251Q
1800  CA 22 1A        CMP   M
1803  3E A9           JNZ   ER8             ;BUMP PNTR'S
1805  BE              INX.  H
1806  C2 22 1A        DCR   C               ;STORE PROGRAM SEGMENT
1809  23              PUSH  H               ;SAVE H,L,B,C
180A  0D              PUSH  B               ;IN RAM,START AT GREG
180B  E5              LXI   B,GREG          ;ADD. OF INST'S
180C  C5              LXI   H,RINST         ;NUMB. OF BYTES
180D  01 77 21        MVI   F,5             ;GET BYTE
1810  21 36 18        MOV   A,M             ;STORE IN RAM
1813  1E 05           STAX  B
1815  7E        V1:   INX   H
1816  02              INX   B               ;BUMP PNTR'S,DCR CNT
1817  23              DCR   E
1818  03              JNZ   V1
1819  1D
181A  C2 15 18        LXI   H,GREG+1        ;STURE PORT #
181D  21 78 21        MOV   M,D             ;IN RAM
1820  72              JMP   GREG            ;OK - TRANSFER
1821  C3 77 21        LXI   H,GREG+2        ;SET UP FOR FLOAT
1824  21 79 21  HOME:  MOV   M,A             ;STORE AWAY INPUT
1827  77              DCX   H
1828  2B              XRA   A               ;ZERO OUT HIGHER BYTES
1829  AF              MOV   M,A             ;BUT CHAR. DOESN'T MATT
182A  77              DCX   H
182B  2B              MOV   M,A
182C  77              CALL  DFXL            ;FLOAT IT
182D  CD DC 0F        LXI   D,GREG          ;FIX D,E RESTORE C,H,L
1830  11 77 21        POP   B
1833  C1              POP   H
1834  E1              RET
1835  C9
1836  0B 00     RINST: IN    0             ;RAM INSTRUCTIONS
1838  C3 24 18        JMP   HOME
183B  CD 14 12  KUNT:  CALL  NUMB            ;NUMBER
183E  DA 47 18        JC    OKK
1841  3E AE           MVI   A,256Q          ;DEC. PNT.?
1843  BE              CMP   M
1844  C2 22 1A        JNZ   ER8
1847  3E 01     OKK:   MVI   A,1             ;MODE=1, IE. INPUT FROM
1849  CD 53 18        CALL  ROKON           ;READ CONSTANT TO GREG
184C  DA 7C 1F        JC    ER9             ;IF ERROR THEN CY=1
184F  11 77 21        LXI   D,GREG          ;PNTS. TO CONSTANT
1852  C9              RET

                ; THIS ROUTINE READS A CONSTANT INTO GREG FROM ASCII
                ; CHARACTERS POINTED TO BY HL AND C
                ; ENTER WITH A=0 => DATA FROM TTY
                ; ENTER WITH A=1 => DATA FROM SOURCE
                ; RETURN WITH CY=1 => ERRUR IN CONVERSION

1853  32 85 21  ROKUN: STA   MODE          ;SAVE MODE FOR ROUT. IN
1856  22 86 21        SHLD  HLINP           ;SAVE HL FUR ROUT. INP
1859  79              MOV   A,C
185A  32 84 21        STA   CREG            ;SAVE C FUR ROUT. INP
185D  21 77 21        LXI   H,GREG          ;WHERE VALUE WILL GO
1860  0E 66           MVI   C,SCH AND 377Q  ;SET UP AND CALL FINPT
1862  CD E8 0F        CALL  FINPT
1865  2A 86 21        LHLD  HLINP           ;RETORE H,L AND C
1868  3A 84 21        LDA   CREG
186B  4F              MOV   C,A
186C  C9              RET                   ;DONE

                ; VAR DECIDES WHETHER A TOKEN IS
                ; A VARIABLE IF SO CY=1 AND
                ; AUDRESS IS COMPUTED,(SUBSCRIPT IS
                ; EVALUATED ETC.), RETURNS WITH DE PNTING
                ; TO VAR. REFERENCED H,L,C,UPDATED
                ; A,B DESTROYED
                ; IF NOT A VARIBLE CY=0
                ; H,L,C ARE LEFT UNTOUCHED

186D  CD 22 12  VAR:   CALL  ALPHA           ;1ST CHAR A LETTER?
1870  00              RNC                   ;NO-NOT VAR.
1871  23              INX   H               ;BUMP PNTR'S
1872  0D              DCR   C
1873  C2 7F 18        JNZ   MORE            ;MORE TO LINE
1876  C5        SC1:  PUSH  B               ;SAVE B,EOL
1877  0E 00           MVI   C,0             ;SET FOR CALL TO FSYM
1879  2B              DCX   H               ;GET SINGLE LETTER
187A  46              MOV   B,M             ;VAR TO B
187B  23              INX   H
187C  C3 81 18        JMP   SCALR
187F  CD 22 12  MORE:  CALL  ALPHA           ;2ND A LETTER?
1882  D2 94 18        JNC   SFSG            ;SO FAR SO GOOD
1885  C5              PUSH  B               ;SAVE C
1886  3E 02           MVI   A,2             ;CHECK FOR DELIMITER
1888  CD F7 12        CALL  SYMSRT
188B  C1              POP   B               ;RESTORE C
188C  3C              INR   A               ;FOUND?
188D  C2 76 18        JNZ   SC1             ;YES
1890  0E        BUPT:  INR   C               ;NOT A VAR.
1891  2B              DCX   H               ;BACK UP PNTR'S
1892  B7              ORA   A               ;CY=0 AND RET
1893  C9              RET
1894  CD 14 12  SFSG:  CALL  NUMB            ;TEST FUR NUMBER
1897  D2 A9 18        JNC   ARCK            ;MAYBE AN ARRAY
189A  23              INX   H               ;ITS A SCALAR
189B  0D              DCP   C               ;BUMP PNTR'S
189C  CA AA 18        JZ    SLOAD           ;EOL
189F  C5              PUSH  B               ;SAVE C
18A0  3E 02           MVI   A,2             ;SET UP FOR SYMSRT
18A2  CD F7 12        CALL  SYMSRT          ;TEST FOR LEGAL
18A5  C1              POP   B               ;GET C BACK
18A6  3C              INR   A               ;DELIMITER FOUND?
18A7  CA 22 1A        JZ    ER8             ;NO, ERROR
18AA  2B        SLOAD: DCX   H               ;MOVE BACK,
18AB  C5              PUSH  B               ;SAVE C
18AC  4E              MOV   C,M             ;GET VAR. INTO
18AD  2B              DCX   H               ;B,C FOR FSYM
18AE  46              MOV   B,M
18AF  23              INX   H
18B0  23              INX   H
18B1  EB        SCALR: XCHG                 ;SAVE H,L IN D,E
18B2  CD 0D 16        CALL  FSYM            ;GET PNTR TO VALUE
18B5  EB              XCHG                  ;RESTORE H,L PNTR TO DE
18B6  C1              POP   B               ;GET C REG BACK
18B7  37              STC                   ;SET CY,RET
18B8  C9              RET
18B9  7E        ARCK:  MOV   A,M             ;ARRAY CHEK, GET CHARAC
18BA  FE A8           CPI   250Q            ;IS IT (?
18BC  CA CD 18        JZ    ARYES           ;YES,IT'S AN ARRAY
18BF  3E 02           MVI   A,2             ;NO-CHEK FOR LEGAL DELI
18C1  C5              PUSH  B               ;SAVE C
18C2  CD F7 12        CALL  SYMSRT
18C5  C1              POP   B               ;RESTORE C
18C6  3C              INR   A               ;DELIMITER FOUND?
18C7  CA 22 1A        JZ    ER8
18CA  C3 76 18        JMP   SC1             ;1 CHAR. SCALAR VAR.
18CD  2B        ARYES: DCX   H               ;YES-WE HAVE ARRAY
18CE  7E              MOV   A,M             ;GET VAR.
18CF  23              INX   H
18D0  F5              PUSH  PSW             ;SAVE VAR.
18D1  CD FB 15        CALL  ICP8            ;BUMP PNTR'S
18D4  CD C7 19        CALL  EVAL            ;EVALUATE SUBSCRIPT
18D7  E5              PUSH  H               ;SAVE REG H,L
18D8  21 7C 21        LXI   H,FREG1
18DB  CD A8 17        CALL  COPDH           ;COPY IT
18DE  EB              XCHG
18DF  E1              POP   H               ;RESTORE H,L
18E0  CD 15 19        CALL  FIX             ;FIX VALUE
18E3  3E A9           MVI   A,251Q          ;CHECK FOR )
18E5  BE              CMP   M
18E6  C2 22 1A        JNZ   ER8
18E9  23              INX   H
18EA  0D              DCR   C               ;BUMP PNTR'S
18EB  13              INX   D               ;PNT TO LOWER 2 BYTES
18EC  13              INX   D
18ED  1A              LDAX  D               ;H-BYTE TO B
18EE  47              MOV   B,A             ;PNT TO LOW BYTE
18EF  13              INX   D               ;LOW BYTE TO A
18F0  1A              LDAX  D               ;KILL CY
18F1  B7              ORA   A               ;START MULT OF OFFSET
18F2  17              RAL                   ;BY 4(BYTES/FLTPT #)
18F3  5F              MOV   E,A             ;GET H BYTE
18F4  78              MOV   A,B
18F5  17              RAL
18F6  57              MOV   D,A             ;DE IS OFFSET*2
18F7  78              MOV   A,E             ;GET LOW
18F8  87              ORA   A               ;KILL CARRY
18F9  17              RAL
18FA  5F              MOV   E,A
18FB  7A              MOV   A,D
18FC  17              RAL
18FD  57              MOV   D,A
18FE  F1              POP   PSW             ;DE CONTAIN OFFSET*4
18FF  C5              PUSH  B               ;GET VAR., SAVE C
1900  4F              MOV   C,A
1901  06 00           MVI   B,0             ;SETUP TO CALL FSYM

1903  E5              PUSH  H               ;SAVE H,L
1904  CD 0D 16        CALL  FSYM            ;GET START ADD.
1907  DA 0F 19        JC    AF(IND)
190A  3E 12           MVI   A,12H           ;ERROR 12
190C  C3 BF 15        JMP   ERROR           ;ARRAY REF. NOT DIM'ED.
190F  19        AF(IND): DAD   D            ;H,L NOW PNT TO START O
1910  EB              XCHG                  ;ARRAY, ADD OFFSET, EXC
1911  F1              POP   H               ;RESTORE PNTR'S AND RET
1912  C1              POP   B
1913  37              STC                   ;SET CY
1914  C9              RET

                ; ROUTINE TO FIX FLOATING POINT
                ; NUMBERS, ALL REG'S BUT A ARE
                ; MAINTAINED. DE PNT TO 4 BYTES
                ; OF # TO BE FIXED

1915  C5        FIX:   PUSH  B
1916  E5              PUSH  H               ;SAVE REG'S
1917  D5              PUSH  D
1918  13              INX   D
1919  13              INX   D               ;PNT TO 4TH BYTE
191A  13              INX   D
191B  1A              LDAX  D
191C  F5              PUSH  PSW             ;SAVE CHAR. (FOR SIGN)
191D  E6 7F           ANI   177Q            ;CHEK IF EXP SIGN IS -
191F  17              RAL
1920  17              RAL
1921  DA 30 19        JC    MINSE
1924  1F              RAR
1925  1F              RAR                   ;RESTORE CHAR
1926  FE 18           CPI   30Q             ;IS IT TOO BIG?
1928  DA 32 19        JC    GOOD
192B  3E 13           MVI   A,13H           ;ERROR 13
192D  C3 BF 15        JMP   ERROR           ;FIX # TOO BIG
1930  1F        MINSE: RAR
1931  1F              RAR
1932  12        GOOD:  STAX  D               ;ABSOLUTE VALUE
1933  1B              DCX   D
1934  1B              DCX   D
1935  1B              DCX   D               ;MOV PNTR BACK
1936  21 7C 21        LXI   H,FREG1
1939  CD A8 17        CALL  COPDH           ;COPY TO FREG1
193C  21 80 21        LXI   H,FREG2         ;STORE .5*2**24 IN
193F  11 6F 19        LXI   D,FDAT          ;FREG2
1942  CD A8 17        CALL  COPDH           ;COPY IT
1945  21 7C 21        LXI   H,FREG1         ;SET UP TO CALL LADD
1948  06 80           MVI   B,FREG2 AND 377Q
194A  0E 66           MVI   C,SCR AND 377Q
194C  CD D0 0F        CALL  LADD            ;ADD THEM,RESULT IN FRE
194F  21 7C 21        LXI   H,FREG1
1952  F1              POP   PSW             ;GET SIGN AND ADD.
1953  D1              POP   D
1954  17              RAL
1955  3E 00           MVI   A,0             ;GET SIGN ONLY
1957  1F              RAR
1958  46              MOV   B,M             ;GET BYTE1
1959  12              STAX  D               ;STORE BYTE 1 OF FIX
195A  78              MOV   A,B
195B  E6 7F           ANI   177Q            ;CLEAR HIGH BIT (FROM A
195D  13              INX   D
195E  23              INX   H
195F  46              MOV   B,M             ;GET BYTE 2
1960  12              STAX  D               ;STORE BYTE 2 OF FIX
1961  13              INX   D
1962  78              MOV   A,B
1963  23              INX   H
1964  46              MOV   B,M             ;GET BYTE 3
1965  12              STAX  D               ;STORE BYTE 3 OF FIX
1966  78              MOV   A,B
1967  13              INX   D
1968  12              STAX  D               ;STURE BYTE 4 OF FIX
1969  1B              DCX   D               ;FIX D PNTR
196A  1B              DCX   D
196B  1B              DCX   D
196C  E1              POP   H
196D  C1              POP   B
196E  C9              RET
196F  80 00 00 18 FDAT: DB   200Q,0,0,30Q

                ; INP SAVES ALL REG'S
                ; SERVES AS A BUFFER ROUTINE BETWEEN FINPT AND
                ; DATA INPUT. IF MODE=0, DATA COMES FROM TTY
                ; IF MODE=1 DATA COMES FROM SOURCE STMTS.
                ; IN ALL CASES, HL,C ARE UPDATED FROM HLINP, AND
                ; CPEG AND RETURNED TO THOSE LOCATIONS

1973  E5        INP:   PUSH  H               ;SAVE ALL REG'S
1974  D5              PUSH  D
1975  C5              PUSH  B
1976  2A 86 21        LHLD  HLINP           ;GET PNTR'S
1979  3A 84 21        LDA   CREG
197C  4F              MOV   C,A
197D  B7              ORA   A               ;CHECK FOR EOL
197E  C2 87 19        JNZ   CHKMD           ;NO CHECK MODE
1981  3E A0     SPACE: MVI   A,240Q          ;SEND A SPACE
1983  C1        IDONE: POP   B               ;RESTORE REG'S
1984  D1              POP   D
1985  E1              POP   H
1986  C9              RET                   ;AND RETURN
1987  3A 85 21  CHKMD: LDA   MODE            ;GET MODE
198A  3D              DCR   A               ;CHECK IT
198B  CA 97 19        JZ    MODE1           ;MODE IS 1
198E  7E              MOV   A,M             ;MODE 0, GET CHAR.
198F  FE AC           CPI   ',' OR 200Q     ;IS IT A ','?
1991  CA 81 19        JZ    SPACE           ;YES - SEND A SPACE
1994  C3 BB 19        JMP   BMPTR           ;NO - SEND IT
1997  CD 14 12  MODE1: CALL  NUMB            ;NUMBER? (ALSO LOADS IT
199A  DA BB 19        JC    BMPTR           ;YES - SEND IT AND BUMP
199D  FE AE           CPI   256Q            ;DEC. PNT.?
199F  CA BB 19        JZ    BMPTR
19A2  FE C5           CPI   305Q            ;E?
19A4  CA BB 19        JZ    BMPTR
19A7  FE AB           CPI   253Q            ;+?
19A9  CA D1 19        JZ    CHEKE
19AC  FE AD           CPI   255Q            ;-?
19AE  C2 81 19        JNZ   SPACE           ;SEND A SPACE
19B1  2B        CHEKE: DCX   H               ;CHEK IF E PRECEDES +,-
19B2  2B              DCX   H               ;BACK UP AND GET PRE-
19B3  7E              MOV   A,M             ;CEDING CHARACTER
19B4  FE C5           CPI   305Q            ;IS IT E?
19B6  C2 81 19        JNZ   SPACE           ;NO,+OR- WAS DELIMITTER
19B9  78              MOV   A,B             ;YES,GET + OR -
19BA  23              INX   H               ;RESTORE H,L
19BB  23        BMPTR: INX   H               ;BUMP AND STORE PNTR'S
19BC  0D              DCR   C
19BD  22 86 21        SHLD  HLINP
19C0  21 84 21        LXI   H,CREG
19C3  71              MOV   M,C
19C4  C3 83 19        JMP   IDONE           ;RESTORE REG'S AND RETJ

                ; THIS ROUTINE WILL EVALUATE UNARY AND/OR
                ; BINARY EXPRESIONS CALLED WITH H AND L
                ; POINTING TO FIRST CHAR. OF EXP., C CONTAINS
                ; NUMBER OF CHAR'S LEFT IN LINE. RETURNS
                ; D(HIGH) AND E(LOW) POINTING TO THE ANSWER
                ; THIS ROUTINE CALLS ITSELF RECURSIVELY
                ; IN ORDER TO EVALUATE SUBSCRIPT
                ; EXPRESIONS.  REG A,H DESTROYED
                ; C,H,L ARE UPCATED

19C7  3E A0     EVAL:  MVI   A,255Q          ;IS IT UNARY -
19C9  BE              CMP   M               ;Z=1 => YES
19CA  F5              PUSH  PSW             ;Z=0 => NC
19CB  C2 D1 19        JNZ   ECAV
19CE  CD FB 15        CALL  ICP8            ;BUMP POINTER
19D1  CD C3 17  ECAV:  CALL  VALUE           ;GET PNTR. TO VALUE
19D4  E5              PUSH  H               ;GET VALUE TO FREG1
19D5  21 7C 21        LXI   H,FREG1
19D8  CD A8 17        CALL  COPDH
19DB  EB              XCHG
19DC  F1              POP   H
19DD  F1              POP   PSW             ;GET SIGN
19DE  C2 EC 19        JNZ   DOL             ;SHALL WE NEGATE?
19E1  13              INX   D               ;YES, POINT TO CHAR.
19E2  13              INX   D
19E3  13              INX   D
19E4  1A              LDAX  D               ;AND LOAD TO A
19E5  17              RAL                   ;ROTATE SIGN TO CY
19E6  3F              CMC                   ;COMPLEMENT IT
19E7  1F              RAR                   ;ROTATE BACK
19E8  12              STAX  D               ;STORE AWAY
19E9  1B              DCX   D               ;AND FIX PNTR.
19EA  1B              DCX   D
19EB  1B              DCX   D
19EC  79        DOL:   MOV   A,C             ;IS THIS END OF LINE?
19ED  B7              ORA   A
19EE  C8              RZ                    ;YES-RETURN
19EF  C5              PUSH  B               ;SAVE C
19F0  3E 02           MVI   A,2             ;NO SET UP TO CALL
19F2  CD F7 12        CALL  SYMSRT          ;SYMSRT AND CALL
19F5  C1              POP   B               ;RESTORE C
19F6  3C              INR   A               ;NO, ERROR
19F7  CA 22 1A        JZ    ER8             ;CHECK FOR EXPRESSION
19FA  D6 0A     ENK:   SUI   10              ;DELIMITER
19FC  D8              RC                    ;SAVE OVERATION
19FD  F5              PUSH  PSW
19FE  CD FB 15        CALL  ICP8            ;BUMP PNTR'S
1A01  B7              ORA   A               ;CLEAR CY
1A02  E5        AGA:   PUSH  H               ;GET BYTES OF NUMBER
```

```
1A03  1A                  LDAX  D              ;AND PLACE ON STACK
1A04  6F                  MOV   L,A
1A05  13                  INX   D
1A06  1A                  LDAX  D
1A07  13                  INX   D
1A08  67                  MOV   H,A            ;2 BYTES TO H,L
1A09  E3                  XTHL                 ;XCHANGE, RESTORES H,L
1A0A  3F                  CMC
1A0B  DA 02 1A            JC    AGA            ;ANOTHER PASS?
1A0E  CD C3 17            CALL  VALUE          ;GET 2ND VALUE
1A11  79                  MOV   A,C            ;CHECK FOR END OF LINE
1A12  B7                  ORA   A              ;IF SO => WELL FORMED
1A13  CA 27 1A            JZ    WFOR
1A16  C5                  PUSH  B              ;SAVE C
1A17  3E 02               MVI   A,2            ;ELSE CALL SYMSRT TO
1A19  CD F7 12            CALL  SYMSRT         ;CHEK FOR EXP. DEL.
1A1C  C1                  POP   B              ;RECOVER IT
1A1D  FE 0A               CPI   10
1A1F  DA 27 1A            JC    WFOR           ;YES, WELL FORMED
1A22  3E 08         ER8:  MVI   A,8            ;ILL-FORMED EXP.
1A24  C3 BE 15            JMP   ERROR
1A27  C5            WFOR: PUSH  B              ;SAVE C, AND H,L
1A28  E5                  PUSH  H
1A29  21 80 21            LXI   H,FREG2        ;COPY 2ND VALUE TO
1A2C  CD A8 17            CALL  COPDH          ;FREG2
1A2F  D1                  POP   D              ;GET BYTES FROM STACK
1A30  C1                  POP   B
1A31  E1                  POP   H              ;INTO FREG1+2
1A32  22 7E 21            SHLD  FREG1+2
1A35  E1                  POP   H              ;AND NEXT 2 BYTES
1A36  22 7C 21            SHLD  FREG1          ;FROM STACK TO FREG1
1A39  EB                  XCHG
1A3A  F1                  POP   PSW            ;GET OPERATION

                    ;
                    ; THIS ROUTINE PERFORMS BINARY OPERATIONS ON OPERANDS IN
                    ; B,C,H,L ARE LEFT UNDISTURBED. A IS DESTROYED
                    ; D,E PNT TO RESULT
                    ; OPERATIONS ARE SPECIFIED BY A REGISTER AS FOLLOWS:
                    ;
                    ;       A=0   =>      FREG1 * FREG2
                    ;       A=1   =>      FREG1 / FREG2
                    ;       A=2   =>      FREG1 + FREG2
                    ;       A=3   =>      FREG1 - FREG2
                    ;
                    ; IN CASE OF ARITHMETIC ERROR A MESSAGE IS SENT TO USER.
                    ; IF A CONTAINS ILLEGAL OPERATION REQUEST ERROR IS SENT T
                    ; (ERROR 8) AND RUN (THE INTERPRETER) IS ABORTED.
                    ;
1A3B  C5            BINOP: PUSH  B              ;SAVE REG'S
1A3C  E5                   PUSH  H
1A3D  21 7C 21             LXI   H,FREG1        ;SET UP PNTR'S TO
1A40  06 80                MVI   B,FREG2 AND 377Q ;FREG'S AND SCR AREA
1A42  0E 6C                MVI   C,SCR AND 377Q ;AND DO OPERATION
1A44  3D                   DCR   A
1A45  FA 87 1A             JM    FMULT          ;0,1=>* OR /
1A48  CA 8D 1A             JZ    DIV            ;2,3=>+ OR -
1A4B  3D                   DCR   A
1A4C  CA 56 1A             JZ    ADDD
1A4F  3D                   DCR   A
1A50  CA 81 1A             JZ    SUBB
1A53  C3 22 1A             JMP   ER8            ;ILLEGAL OPER.
1A56  CD D0 0F      ADDD:  CALL  LADD           ;DO ADDITION
1A59  54            ASBC:  MOV   D,H            ;FIX PNTR'S FOR RET.
1A5A  5D                   MOV   E,L
1A5B  B7            FPEKR: ORA   A              ;SET FLAGS
1A5C  CA 7E 1A             JZ    NFPEP          ;NO ERROR
1A5F  D5                   PUSH  D              ;SAVE DE
1A60  F5                   PUSH  PSW            ;SAVE A
1A61  CD D0 12             CALL  WRIT           ;DUMP BUFFER
1A64  F1                   POP   PSW            ;GET A BACK
1A65  21 77 1A             LXI   H,WFPER        ;RETURN ADDRESS
1A68  E5                   PUSH  H              ;SAVE ON STACK
1A69  21 5A 14             LXI   H,ODATA        ;MESSAGE TABLE
1A6C  17                   RAL                  ;UNDERFLOW?
1A6D  DA 3F 14             JC    FOR12          ;YES
1A70  17                   RAL                  ;OVERFLOW?
1A71  DA 40 14             JC    FOR11          ;YES
1A74  C3 41 14             JMP   FOR13          ;NO - ITS ZERODIVIDE
1A77  21 5A 14      WFPER: LXI   H,ODATA        ;MESSAGE TABLE
1A7A  CD 0F 15             CALL  ERLN           ;PRINT 'IN LINE --' (US
1A7D  D1                   POP   D              ;RESTORE REG'S
1A7E  E1            NFPER: POP   H
1A7F  C1                   POP   B
1A80  C9                   RET
1A81  CD D9 0F      SUBB:  CALL  LSUB           ;DO SUBTRACTION
1A84  C3 59 1A             JMP   ASBC
1A87  CD D3 0F      FMULT: CALL  LMUL           ;DO MULT.
1A8A  C3 90 1A             JMP   MDBC
1A8D  CD D6 0F      DIV:   CALL  LDIV           ;DO DIV.
1A90  54            MDBC:  MOV   D,H            ;AND FIX PNTR'S FOR RET
1A91  59                   MOV   E,C
1A92  C3 5B 1A             JMP   FPERR          ;CHECK FOR ERROR

                    ;
                    ; PRINT PROCESSOR
                    ;
1A95  2A 5B 21      PRI:   LHLD  CPNT
1A98  23                   INX   H              ;INCR. PAST KEYWORD
1A99  23                   INX   H
1A9A  23                   INX   H
1A9B  CD F6 15             CALL  ICP7
1A9E  23                   INX   H              ;BUMP PNTRS
1A9F  0D                   DCR   C
1AA0  06 00                MVI   B,0            ;SET CHAR CNT
1AA2  C2 AE 1A             JNZ   PLOOP          ;CONTINUE IF MORE
1AA5  04                   INR   B              ;NOTHING MORE, PAD A NU
1AA6  3E 00                MVI   A,0
1AA8  CD AD 12             CALL  PAD
1AAB  C3 46 1B             JMP   PEND           ;WRITE IT AND CONTINUE
1AAE  7E            PLOOP: MOV   A,M            ;GET CHARACTER
1AAF  FE A2                CPI   '"'+200Q       ;IS IT "?
1AB1  C2 D6 1A             JNZ   EXPRE          ;NO
1AB4  CD F6 15      QUOTE: CALL  ICP7           ;GET CHARACTER TO A
1AB7  7E                   MOV   A,M
1AB8  FE A2                CPI   '"'+200Q       ;IS IT "?
1ABA  CA C8 1A             JZ    QCHEK
1ABD  04            QOTOK: INR   B              ;INCREMENT CNT
1ABE  50                   MOV   D,B            ;SAVE IN D
1ABF  06 01                MVI   B,1            ;PAD ONCE
1AC1  CD AD 12             CALL  PAD
1AC4  42                   MOV   B,D            ;RESTURE CNT
1AC5  C3 B4 1A             JMP   AGAIN          ;QUOTH
1AC8  23            QCHEK: INX   H              ;BUMP PNTRS
1AC9  0D                   DCR   C
1ACA  CA 46 1B             JZ    PEND           ;EOL
1ACD  7E                   MOV   A,M
1ACE  FE A2                CPI   '"'+200Q       ;ANOTHER "?
1AD0  CA BD 1A             JZ    QOTOK
1AD3  C3 07 1B             JMP   SCOLN
1AD6  CD 22 12      EXPRE: CALL  ALPHA          ;IS IT A LETTER
1AD9  DA ED 1A             JC    PRTIT          ;YES, EVALUATE AND PRIN
1ADC  CD 14 12             CALL  NUMB           ;IS IT A NUMB?
1ADF  DA ED 1A             JC    PRTIT          ;YES, EVALUATE AND PRIN
1AE2  7E                   MOV   A,M
1AE3  FE AE                CPI   '.'+200Q       ;IS IT A DECIMAL PNT?
1AE5  CA ED 1A             JZ    PRTIT          ;YES EVALUATE, PRINT
1AE8  FE AD                CPI   '-'+200Q       ;IS IT A -?
1AEA  C2 07 1B             JNZ   SCOLN          ;NO, CHECK FOR ;
1AED  C5            PRTIT: PUSH  B              ;SAVE CNT
1AEE  CD C7 19             CALL  EVAL           ;EVALUATE EXPRESION
1AF1  C5                   PUSH  B              ;SAVE C,H,L
1AF2  E5                   PUSH  H
1AF3  EB                   XCHG                 ;DE TO HL
1AF4  0E 66                MVI   C,SCR AND 377Q ;SET UP, CONVERT
1AF6  CD E5 0F             CALL  CONV
1AF9  E1                   POP   H              ;RESTORE REG'S
1AFA  C1                   POP   B
1AFB  79                   MOV   A,C
1AFC  C1                   POP   B
1AFD  4F                   MOV   C,A
1AFE  B7                   ORA   A              ;CHECK EOL
1AFF  CA 46 1B             JZ    PEND
1B02  3E 0B                MVI   A,11           ;UPDATE CNTR
1B04  80                   ADD   B
1B05  47                   MOV   B,A
1B06  7E            SCOLN: MOV   A,M            ;GET CHAR.
1B07  FE BB                CPI   ';'+200Q       ;IS IT ;?
1B09  CA 39 1B             JZ    SONWD          ;YES
1B0C  FE AC                CPI   ','+200Q       ;IS IT ,?
1B0E  C2 2C 17             JNZ   ER6            ;NO-UNEXPECTED CHAR.
1B11  AF                   XRA   A              ;ZERO A
1B12  C6 0D         ADFLD: ADI   13             ;ADD FIELD LENGTH
1B14  B8                   CMP   B              ;COMPARE TO CNT
1B15  CA 1B 1B             JZ    $+6
1B18  D2 2D 1B             JNC   FLDFD
1B1B  FE 34                CPI   52             ;LAST FLD?
1B1D  C2 12 1B             JNZ   ADFLD
1B20  CD D0 12      ONWD:  CALL  WRIT           ;YES-WRITE LINE
1B23  06 00                MVI   B,0            ;RESET CNT
1B25  23                   INX   H              ;BUMP PNTRS
1B26  0D                   DCR   C
1B27  CA 46 1B             JZ    PEND
1B2A  C3 AE 1A             JMP   PLOOP
1B2D  90            FLDFD: SUB   B              ;FOUND FIELD
1B2E  50                   MOV   D,B            ;DETERMIN  OF SPACES T
1B2F  5F                   MOV   E,A            ;SET UP TO CALL PAD
1B30  47                   MOV   B,A
1B31  3E A0                MVI   A,240Q
1B33  CD AD 12             CALL  PAD            ;PAD SPACES
1B36  7A                   MOV   A,D
1B37  83                   ADD   E              ;NEW CNT
1B38  47                   MOV   B,A            ;SAVE IN B
1B39  23            SONWD: INX   H              ;CHECK EOL
1B3A  0D                   DCR   C
1B3B  C2 AE 1A             JNZ   PLOOP

1B3E  16 01                MVI   D,1            ;SUPPRESS CR/LF
1B40  CD D2 12             CALL  WRIT1
1B43  C3 49 1B             JMP   $+6
1B46  CD D0 12      PEND:  CALL  WRIT           ;DUMP BUFFER, CONTINUE
1B49  C3 BC 1B             JMP   IEND

                    ;
                    ; INPUT PROCESSOR - READS VALUES FROM TTY
                    ; THEY MUST BE DELIMITED BY COMMAS ONLY
                    ;
1B4C  79            INPUT: MOV   A,C            ;IN CASE OF ERROR
1B4D  32 5F 21             STA   PL6            ;SAVE
1B50  2A 5B 21      INPER: LHLD  CPNT           ;INPUT LINE (V-STRING)
1B53  23                   INX   H              ;ADJUST PNTR'S
1B54  23                   INX   H
1B55  23                   INX   H
1B56  CD F6 15             CALL  ICP7
1B59  CD F6 15             CALL  ICP7
1B5C  C5            PRMPT: PUSH  B              ;SAVE PNTR'S
1B5D  E5                   PUSH  H
1B5E  06 01                MVI   B,1            ;SEND PROMPT
1B60  3E 3A                MVI   A,':'
1B62  50                   MOV   D,B            ;TO SUPPRESS CR/LF
1B63  CD AD 12             CALL  PAD            ;PAD IT
1B66  CD D2 12             CALL  WRIT1          ;WRITE IT
1B69  21 81 21             LXI   H,IBUF         ;ADD. OF INPUT BUFFER
1B6C  CD C9 13             CALL  TTYIN          ;READ A LINE
1B6F  EB                   XCHG                 ;ADD. OF K-STRING TO DE
1B70  E1                   POP   H              ;ADD. OF V-STRING
1B71  C1                   POP   B              ;V-STRING CNT TO C
1B72  47                   MOV   B,A            ;K-STRING CNT TO B
1B73  CD 92 1B             CALL  STRIN          ;TRANSFER CONSTANTS TO
1B76  CA 89 1B             JZ    INPOK          ;NO ERROR
1B79  21 5A 14             LXI   H,ODATA        ;SEND ERROR MESSAGE
1B7C  CD 42 14             CALL  FORM9
1B7F  CD D0 12             CALL  WRIT
1B82  3A 5F 21             LDA   PL6            ;GET V-STRING CNT
1B85  4F                   MOV   C,A
1B86  C3 50 1B             JMP   INPER          ;START AGAIN
1B89  DA 5C 1B      INPOK: JC    PRMPT          ;NEED MORE CONSTANTS
1B8C  2A 5D 21      IEND:  LHLD  KFPNT          ;ALL OK - GET NEW PNTR.
1B8F  C3 BC 16             JMP   ILOOP          ;CONTINUE

                    ;
                    ; THIS ROUTINE TRANSFERS THE FLOATING POINT VALUES
                    ; OF AN ASCII STRING OF CONSTANTS TO THE LOCATIONS
                    ; SPECIFIED BY AN ASCII STRING OF VARIBLES
                    ; POINTER AND LINE CNT OF VAR. STRING ARE IN HL,C
                    ; POINTER AND LINE CNT OF CONST. STRING ARE IN DE,B
                    ; ON RETURN:
                    ;
                    ;       Z=0 AND CY=0    ALL OK
                    ;       Z=0 AND CY=1    NEED MORE CONSTANTS
                    ;       Z=1             ERROR IN CONVERSION
                    ;
                    ; ALL POINTERS AND LINE CNT'S ARE RETURNED UPDATED
                    ;
1B92  79            STRIN: MOV   A,C            ;GET V-STRING CNT
1B93  B7                   ORA   A              ;TEST FOR EOL
1B94  C8                   RZ                   ;DONE, CY=0 => ALL OK
1B95  7E                   MOV   A,M            ;GET CHAR.
1B96  FE AC                CPI   ','+200Q       ;IS IT A ,?
1B98  C2 A0 1B             JNZ   STOKV          ;IT'S NOT A ,
1B9B  23                   INX   H              ;COMMA, BUMP PNTR'S
1B9C  0D                   DCR   C
1B9D  CA C8 1B             JZ    ERRET          ;POSSIBLE ERROR (IF EOL
1BA0  78            STOKV: MOV   A,B            ;GET K-STRING LENGTH
1BA1  B7                   ORA   A              ;TEST FOR EOL
1BA2  37                   STC                  ;IN CASE IT'S EOL
1BA3  C8                   RZ                   ;RET, CY=1 => NEED MORE
1BA4  1A                   LDAX  D              ;GET CHAR
1BA5  FE AC                CPI   ','+200Q       ;TEST FOR ,
1BA7  C2 AF 1B             JNZ   STOKK          ;NOT A ,  - READY TO GO
1BAA  13                   INX   D              ;BUMP PNTR'S
1BAB  05                   DCR   B
1BAC  CA C8 1B             JZ    ERRET          ;POSSIBLE ERROR (IF EOL
1BAF  C5            STOKK: PUSH  B              ;SAVE K-STRING CNT
1BB0  D5                   PUSH  D              ;SAVE K-STRING PNTR
1BB1  CD 0D 18             CALL  VAR            ;ADD. TO VARIBLE TO DE
1BB4  EB                   XCHG                 ;VAR. ADD TO H,L
1BB5  22 8A 21             SHLD  VARAD          ;SAVE
1BB8  E1                   POP   H              ;ADDRESS OF K-STRING
1BB9  79                   MOV   A,C            ;V-STRING CNT TO A
1BBA  C1                   POP   B              ;K-STRING CNT TO B
1BBB  48                   MOV   C,B            ;K-STRING CNT TO C
1BBC  F5                   PUSH  PSW            ;SAVE V-STRING CNT
1BBD  D5                   PUSH  D              ;SAVE V-STRING ADD.
1BBE  3E 00                MVI   A,0            ;A=0 => DATA FROM TTY
1BC0  CD 53 18             CALL  PDXCN          ;GET CONSTANT TO GREG
1BC3  D2 CB 1B             JNC   STNEP
1BC6  E1                   POP   H              ;EMPTY STACK
1BC7  F1                   POP   H
1BC8  AF            ERRET: XRA   A              ;ERROR
1BC9  3C                   INR   A
1BCA  C9                   RET
1BCB  E5            STNER: PUSH  H              ;SAVE K-STRING PNTR.
1BCC  2A 8A 21             LHLD  VARAD          ;GET VAR. ADD
1BCF  11 77 21             LXI   D,GREG         ;ADD. TO CONST.
1BD2  CD A8 17             CALL  COPDH          ;COPY IT TO VARIABLE LO
1BD5  D1                   POP   D              ;K-STRING PNTR  TO DE
1BD6  41                   MOV   B,C            ;K-STRING LENGTH TO B
1BD7  E1                   POP   H              ;V-STRING PNTR. TO HL
1BD8  F1                   POP   PSW            ;V-STRING LENGTH TO C
1BD9  4F                   MOV   C,A
1BDA  C3 92 1B             JMP   STRIN          ;LOOP

                    ;
                    ; LET STMT. PROCESSOR
                    ;
1BDD  2A 5B 21      LET:   LHLD  CPNT           ;GET PNTR.
1BE0  23                   INX   H              ;FIX PNTR.
1BE1  23                   INX   H
1BE2  23                   INX   H
1BE3  79                   MOV   A,C            ;CHECK FOR EOL
1BE4  B7                   ORA   A
1BE5  C2 ED 1B             JNZ   LOK
1BE8  3E 07         ER7:   MVI   A,7
1BEA  C3 BE 15             JMP   ERROR
1BED  CD 6D 18      LOK:   CALL  VAR            ;GET ADDRESS TO VAR.
1BF0  DA 32 1C             JC    SAVV           ;IT'S A VARIABLE
1BF3  3E 03                MVI   A,3            ;NO-CHEK FOR FUNC.
1BF5  CD F7 12             CALL  SYMSRT
1BF8  FF FF                CPI   377Q
1BFA  CA 22 1A             JZ    ER8
1BFD  3D                   DCR   A
1BFE  C2 78 1C             JNZ   ER10           ;ILLEGAL USE OF FUNC.
1C01  23                   INX   H              ;IT'S PUT,UPDATE H,L
1C02  23                   INX   H
1C03  23                   INX   H
1C04  79                   MOV   A,C            ;EOL CHK
1C05  B7                   ORA   A
1C06  CA 22 1A             JZ    ER8
1C09  7E                   MOV   A,M            ;CHEK FOR (
1C0A  FE A8                CPI   250Q
1C0C  C2 22 1A             JNZ   ER8
1C0F  CD FB 15             CALL  ICP8           ;BUMP PNTRS
1C12  CD C7 19             CALL  EVAL           ;EVALUATE AND FIX
1C15  E5                   PUSH  H              ;SAVE H,L
1C16  21 7C 21             LXI   H,FREG1
1C19  CD A8 17             CALL  COPDH          ;COPY IT
1C1C  EB                   XCHG
1C1D  E1                   POP   H
1C1E  CD 15 19             CALL  FIX
1C21  13                   INX   D
1C22  13                   INX   D
1C23  13                   INX   D
1C24  1A                   LDAX  D              ;GET LOWEST BYTE
1C25  F5                   PUSH  PSW            ;PORT # IS SAVED
1C26  7E                   MOV   A,M
1C27  FE A9                CPI   251Q           ;CHECK FOR )
1C29  C2 22 1A             JNZ   ER8
1C2C  CD FB 15             CALL  ICP8           ;BUMP PNTR'S
1C2F  16 FF                MVI   D,377Q
1C31  5A                   MOV   E,D
1C32  D5            SAVV:  PUSH  D              ;KEEP ADDRESS
1C33  7E                   MOV   A,M            ;CHEK FOR =
1C34  FE BD                CPI   275Q
1C36  C2 22 1A             JNZ   ER8
1C39  CD FB 15             CALL  ICP8           ;BUMP PNTRS
1C3C  CD C7 19             CALL  EVAL           ;EVALUATE EXPRESSION
1C3F  E1                   POP   H              ;GET ADDRESS
1C40  CD A3 12             CALL  CHK1
1C43  DA 4C 1C             JC    PTFIN          ;IT WAS A PUT
1C46  CD A8 17             CALL  COPDH          ;COPY TO ADDRESS
1C49  C3 BC 1B             JMP   IEND           ;CONTINUE
1C4C  21 7C 21      PTFIN: LXI   H,FREG1        ;COPY VALUE TO FREG1
1C4F  CD A8 17             CALL  COPDH
1C52  EB                   XCHG
1C53  CD 15 19             CALL  FIX            ;FIX THE VALUE
1C56  13                   INX   D
1C57  13                   INX   D
1C58  13                   INX   D
1C59  1A                   LDAX  D
1C5A  4F                   MOV   C,A            ;SAVE IN C
1C5B  21 73 1C             LXI   H,PTLST        ;ADD OF BYTES TO GO TO
1C5E  11 77 21             LXI   D,GREG         ;RAM AT GREG
1C61  06 05                MVI   B,5            ;BYTE CNT
1C63  7E            PRII:  MOV   A,M            ;STORE PROG. SEG. IN
1C64  12                   STAX  D              ;RAM
1C65  23                   INX   H
1C66  13                   INX   D
1C67  05                   DCR   B
1C68  C2 63 1C             JNZ   PRII
1C6B  F1                   POP   PSW            ;GET PORT #
1C6C  21 7D 21             LXI   H,GREG+1
1C6F  77                   MOV   M,A            ;STORE
1C70  79                   MOV   A,C            ;GET DATA OUT TO A
1C71  28                   DCX   H              ;TRANSFER
```

```
1C72  F9                  PCHL
1C73  D3 00       PINST:  OUT     0               ;RAM INSTRUCTIONS
1C75  C3 BC 1B            JMP     IEND
1C78  3E 10       ER10:   MVI     A,10H
1C7A  C3 BE 15            JMP     ERROR
                  ;
                  ; IF STMT. PROCESSOR
                  ;
1C7D  2A 56 21    IFRT:   LHLD    CPNT            ;GET PNTR., ADJUST
1C80  23                  INX     H
1C81  0E 00               MVI     C,0             ;CHECK EOL
1C83  CD F6 15            CALL    ICP7
1C86  CD C7 19            CALL    EVAL            ;EVALUATE EXPRESSION
1C89  79                  MOV     A,C
1C8A  B7                  ORA     A               ;CHECK EOL
1C8B  CA E5 1B            JZ      ER7
1C8E  E5          IAGA:   PUSH    H               ;SAVE H,L, PUT VALUE ON
1C8F  1A                  LDAX    D
1C90  6F                  MOV     L,A
1C91  1A                  LDAX    D
1C92  13                  INX     D
1C93  67                  MOV     H,A
1C94  E3                  XTHL                    ;RESTORE H,L
1C95  3F                  CMC
1C96  DA 8D 1C            JC      IAGA            ;ANOTHER PASS?
1C99  3E 02               MVI     A,2
1C9B  CD F7 12            CALL    SYMSRT          ;CHEK TYPE OF RELATION
1C9E  FE 04               CPI     4               ;WAS IT LEGAL?
1CA0  DA A8 1C            JC      III
1CA3  3E 14       ER14:   MVI     A,14H
1CA5  C3 BE 15            JMP     ERROR
1CA8  FE 02       III:    CPI     2               ;WAS IT A ,?
1CAA  CA A3 1C            JZ      ER14
1CAD  3C                  INR     A               ;ALL OK, INC,SAVE
1CAE  F5                  PUSH    PSW
1CAF  0C                  INR     C
1CB0  CD F6 15            CALL    ICP7            ;BUMP PNTRS
1CB3  3E 02               MVI     A,2             ;CALL SYMSRT
1CB5  CD F7 12            CALL    SYMSRT
1CB8  FE FF               CPI     377Q            ;FOUND ANYTHING?
1CBA  CA D5 1C            JZ      RELAT           ;DONE
1CBD  FE 02               CPI     2
1CBF  CA A3 1C            JZ      ER14            ;IT WAS A ,
1CC2  FE 04               CPI     4
1CC4  D2 A3 1C            JNC     ER14            ;NOT LEGAL
1CC7  3C                  INR     A
1CC8  47                  MOV     B,A
1CC9  0C                  INR     C
1CCA  CD F6 15            CALL    ICP7
1CCD  F1                  POP     PSW             ;GET SECOND RELATION
1CCE  80                  ADD     B               ;ADD THEM
1CCF  F5                  PUSH    PSW             ;AND SAVE
1CD0  FE 08               CPI     10Q             ;TEST FOR ==
1CD2  CA A3 1C            JZ      ER14

                  ; AT THIS POINT:
                  ; RELATION IS STORED ON TOP OF STACK (PUSH PSW) ACCORDING
                  ; THE FOLLOWING
                  ;
                  ;       1 =>  <
                  ;       2 =>  >
                  ;       3 =>  <>
                  ;       4 =>  =
                  ;       5 =>  <=
                  ;       6 =>  >=
                  ;
1CD5  CD C7 19    RELAT:  CALL    EVAL            ;EVALUATE
1CD8  E5                  PUSH    H               ;SAVE H,L
1CD9  21 80 21            LXI     H,FREG2         ;COPY TO FREG2
1CDC  CD A8 17            CALL    COPDH
1CDF  E1                  POP     H               ;GET H,L
1CE0  F1                  POP     PSW             ;AND RELATION
1CE1  E3                  XTHL                    ;GET 2ND 2 BYTES
1CE2  22 7E 21            SHLD    FREG1+2         ;STORE
1CE5  E1                  POP     H               ;GET 1ST 2 BYTES,STORE
1CE6  E3                  XTHL
1CE7  22 7C 21            SHLD    FREG1
1CEA  C5                  PUSH    B
1CEB  F5                  PUSH    PSW             ;SAVE A,B,C
1CEC  CD 2D 1D            CALL    FCOMP           ;COMPARE NUMBERS
1CEF  57                  MOV     D,A             ;SAVE RESULT IN D
1CF0  F1                  POP     PSW             ;GET RELATION,B,C
1CF1  C1                  POP     B
1CF2  BA                  CMP     D               ;SAME?
1CF3  CA 16 1D            JZ      TRUE            ;YES
1CF6  D6 04               SUI     4
1CF8  F2 08 1D            JP      NOT3            ;NOT RELATION 3
1CFB  3C                  INR     A               ;IS IT RELATION 3?
1CFC  C2 12 1D            JNZ     FALSE           ;NO, ITS FALSE
1CFF  3E 04               MVI     A,4             ;IT IS, CHECK FOR INEQU
1D01  BA                  CMP     D
1D02  C2 16 1D            JNZ     TRUE
1D05  C3 12 1D            JMP     FALSE
1D08  BA          NOT3:   CMP     D               ;RELATION 5,6 TRUE?
1D09  CA 16 1D            JZ      TRUE            ;YES
1D0C  3E 04               MVI     A,4             ;IT WAS, CHECK FOR EQUA
1D0E  BA                  CMP     D
1D0F  CA 16 1D            JZ      TRUE
1D12  E1          FALSE:  POP     H               ;CONTINUE
1D13  C3 BC 1B            JMP     IEND
1D16  E1          TRUE:   POP     H
1D17  06 04               MVI     B,4
1D19  CD F6 15    THEN:   CALL    ICP7            ;INCREMENT PAST THEN
1D1C  05                  DCR     B
1D1D  C2 19 1D            JNZ     THEN
1D20  C3 09 17            JMP     GTRA            ;TRANSFER TO GOTO

                  ; ROUTINE FCOMP COMPARES 2 FLOATING POINT #'S.  THEY ARE
                  ; TO BE IN FREG1 AND FREG2.
                  ; ALL REGISTERS ARE DESTROYED.
                  ; THE VALUE RETURNED IN REG A IS RESULT OF COMPARISON.
                  ; RESULTS ARE AS FOLLOWS:
                  ;
                  ;       A=1  =>     FREG1 < FREG2
                  ;       A=2  =>     FREG1 > FREG2
                  ;       A=4  =>     FREG1 = FREG2
                  ;
1D23  21 7E 21    FCOMP:  LXI     H,FREG1+3       ;PNTS TO CHAR OF 1ST
1D26  11 83 21            LXI     D,FREG2+3       ;PNTS TO CHAR OF 2ND
1D29  7E                  MOV     A,M             ;GET  1 CHAR
1D2A  06 80               MVI     B,200Q          ;MASK TO B
1D2C  A0                  ANA     B               ;GET SIGN,  1
1D2D  4F                  MOV     C,A             ;SAVE IN C
1D2E  1A                  LDAX    D               ;GET CHAR  2
1D2F  A0                  ANA     B               ;GET SIGN  2
1D30  A9                  XRA     C
1D31  CA 3B 1D            JZ      SINEQ           ;SAME SIGNS
1D34  79                  MOV     A,C             ;OPPISITE SIGNS,GET  1
1D35  17                  RAL                     ;ROTATE TO CY
1D36  3E 01               MVI     A,1
1D38  D8                  RC                      ;FREG1 < FREG2 => A=1
1D39  3C                  INR     A               ;ELSE FREG1 > FREG2
1D3A  C9                  RET                     ;AND A=2
1D3B  C5          SINEQ:  PUSH    B               ;SAVE SIGN
1D3C  2B                  DCX     H               ;PNTR TO  1 IN H,L
1D3D  2B                  DCX     H.
1D3E  2B                  DCX     H
1D3F  11                  MOV     H,E             ;PNTR TO  2 IN B
1D40  05                  DCR     B
1D41  05                  DCR     B
1D42  05                  DCR     B
1D43  CD DF 1F            CALL    LRCM            ;COMPARE MAGNITUDES
                  ; AT THIS POINT Z=1 => =, CY=1 => <2
1D46  C1                  POP     B               ;GET SIGN BACK
1D47  C2 4D 1D            JNZ     $+6
1D4A  3E 04               MVI     A,4             ;EQUAL => A=4
1D4C  C9                  RET
1D4D  79                  MOV     A,C             ;GET SIGN TO A
1D4E  3C                  INR     A               ;SET SIGN BIT
1D4F  3F 01               MVI     A,1
1D51  FA 57 1D            JM      $+6             ;SIGN IS NEGATIVE
1D54  D8                  RC                      ;SIGN=+ AND A<5(FREG1)<
1D55  3C                  INR     A               ;ABS(FREG1)>ABS(FREG2)
1D56  C9                  RET
1D57  D0                  RNC                     ;SIGN=- AND ABS(FREG1)>
1D58  3C                  INR     A               ;ABS(FREG1)<ABS(FREG2)
1D59  C9                  RET
                  ;
                  ; CALL PROCESSOR
                  ;
1D5A  21 BC 1B    CALLP:  LXI     H,IEND          ;INIT RETURN ADDRESS
1D5D  E5                  PUSH    H
1D5E  2A 56 21            LHLD    CPNT            ;INIT PLINTERS
1D61  23                  INX     H
1D62  23                  INX     H
1D63  23                  INX     H
1D64  CD F6 15            CALL    ICP7
1D67  7E                  MOV     A,M             ;GET CHAP
1D68  FE A8               CPI     '('+200Q        ;IS IT A (?
1D6A  C2 ER 1B            JNZ     ER7             ;NAU
1D6D  CD F6 15            CALL    ICP7            ;BUMP PNTRS
1D70  CD 2A 12            CALL    CVB             ;GET SUB
1D73  85                  ADD     L               ;UPDATA H,L
1D74  6F                  MOV     L,A
1D75  3E 00               MVI     A,0
1D77  8C                  ADC     H
1D78  67                  MOV     H,A             ;D NOW CONTAINS SUB
1D79  E5                  PUSH    H               ;SAVE HL
1D7A  21 AC 21            LXI     H,SUBS          ;GET START OF SUB TABLE
1D7D  7E          NOSUB:  MOV     A,M             ;GET ENTRY
1D7E  BA                  CMP     D               ;COMPARE
1D7F  CA 8E 1D            JZ      FNDSB           ;FOUND IT
1D82  23                  INX     H               ;PNT TO NEXT ENTRY
1D83  23                  INX     H
1D84  23                  INX     H
1D85  3C                  INR     A               ;CHECK TO SEE IF LAST S
1D86  C2 7E 1D            JNZ     NOSUB
1D89  3F 15               MVI     A,15H           ;ER 15 - NO SUB BY THIS
1D8B  C3 BE 15            JMP     ERROR
1D8E  23          FNDSB:  INX     H               ;FOUND IT,GET STARTING
1D8F  5E                  MOV     E,M
1D90  23                  INX     H
1D91  56                  MOV     H,M
1D92  6B                  MOV     L,E             ;AND SAVE IT
1D93  22 50 21            SHLD    SBSAV
1D96  2A 54 21            LHLD    NXTSP           ;INIT MEMORY SCRATCH AR
1D99  22 88 21            SHLD    MESCR
1D9C  E1                  POP     H               ;GET SOURCE PNTR BACK
1D9D  7E          PARLP:  MOV     A,M             ;GET CHAR
1D9E  FE A5               CPI     ')'+200Q        ;IS IT )?
1DA0  CA CC 1D            JZ      CLSUB           ;YES - GO CALL SUB
1DA3  FE AC               CPI     ','+200Q        ;DO WE HAVE A ,?
1DA5  C2 2C 17            JNZ     ER6             ;UNEXPECTED CHARACTER
1DA8  CD F6 15            CALL    ICP7            ;BUMP PNTRS
1DAB  CD 6E 18            CALL    VAR             ;DO WE HAVE A VARIABLE
1DAE  D2 B5 1D            JNC     PREXP           ;NO
1DB1  D5                  PUSH    D               ;YES - SAVE ADDRESS
1DB2  C3 9D 1D            JMP     PARLP           ;CONTINUE
1DB5  CD C7 19    PREXP:  CALL    EVAL            ;EVALUATE EXPRESSION
1DB8  E5                  PUSH    H               ;SAVE H,L
1DB9  2A 88 21            LHLD    MESCR           ;GET SCRATCH AREA
1DBC  CD A6 17            CALL    COPDH           ;COPY TO IT
1DBF  D1                  POP     D               ;HL TO DE
1DC0  E5                  PUSH    H               ;SAVE ADDRESS
1DC1  23                  INX     H               ;UPDATE MESCR
1DC2  23                  INX     H
1DC3  23                  INX     H
1DC4  23                  INX     H
1DC5  22 88 21            SHLD    MESCR           ;SAVE IT
1DC8  EB                  XCHG                    ;GET H,L BACK
1DC9  C3 9D 1D            JMP     PARLP           ;CONTINUE
1DCC  2A 50 21    CLSUB:  LHLD    SBSAV           ;START OF POUTINE
1DCF  F9                  PCHL                    ;TRANSFER
                  ;
                  ; GOSUB PROCESSOR
                  ;
1DD0  21 2C 1C    GOSUB:  LXI     H,ILOOP         ;FOR RETURN STMT.
1DD3  E5                  PUSH    H               ;TO STACK
1DD4  2A 56 21            LHLD    KFPNT           ;PNTR. TO NEXT STMT.
1DD7  E5                  PUSH    H               ;SAVE ON STACK
1DD8  2A 54 21            LHLD    NXTSP           ;CHECK MEMORY
1DDB  CD 14 15            CALL    MEMFUL
1DDE  2A 56 21            LHLD    CPNT            ;GET CHAR. PNTR
1DE1  23                  INX     H
1DE2  C3 03 17            JMP     USENT           ;PART OF GOTO TO FINISH
                  ;
                  ; RETURN STMT. PROCESSOR
                  ;
1DE5  F1          RETRN:  POP     H               ;GET RETURN ADD. FROM S
1DE6  C9                  RET                     ;CONTINUE
                  ;
                  ; FOR STATEMENT PROCESSOR
                  ;
1DE7  2A 56 21    FOR:    LHLD    CPNT            ;FIX PNTRS
1DEA  0C                  INR     C
1DEB  23                  INX     H
1DEC  23                  INX     H
1DED  CD F6 15            CALL    ICP7
1DF0  CD 42 12            CALL    ALPHA           ;LETTER?
1DF3  D2 9A 1F            JNC     ER21            ;NO
1DF6  46                  MOV     B,M             ;GET IT TO B
1DF7  CD F6 15            CALL    ICP7            ;BUMP PNTR'S
1DFA  51                  MOV     D,C             ;SAVE C
1DFB  0E 00               MVI     C,0             ;INIT C TO 0
1DFD  CD 14 12            CALL    NUMB            ;NUMBER?
1E00  D2 09 1E            JNC     $+9             ;NE
1E03  4F                  MOV     C,M             ;YES, GET IT
1E04  23                  INX     H               ;BUMP PNTR'S
1E05  15                  DCR     D
1E06  CA E8 19            JZ      ER7             ;PREMATURE EOL
1E09  E5                  PUSH    H               ;SAVE H,L
1E0A  CD 0D 16            CALL    FSYM            ;GET VAR. LOCATION
1E0D  E3                  XTHL                    ;PUT ON STACK, GET H,L
1E0E  59                  MOV     E,C             ;VARIABLE TO D,E
1E0F  4A                  MOV     C,D             ;RESTORE C
1E10  50                  MOV     D,B
1E11  EB                  XCHG                    ;SAVE VAR NAME
1E12  22 8C 21            SHLD    VNAME
1E15  EB                  XCHG                    ;RESTORE H,L
1E16  7E                  MOV     A,M             ;LOOK FOR =
1E17  FE BD               CPI     '=' OR 200Q
1E19  C2 B1 1F            JNZ     ER16
1E1C  CD F6 15            CALL    ICP7            ;BUMP PNTR'S
1E1F  CD C7 19            CALL    EVAL            ;EVALUATE EXPRESSION
1E22  E3                  XTHL                    ;VARIABLE LOCATION
1E23  CD A8 17            CALL    COPDH           ;WRITE VALUE
1E26  22 8E 21            SHLD    VLOC            ;SAVE PNTR TO VARIABLE
1E29  E1                  POP     H               ;GET H,L BACK
1E2A  79                  MOV     A,C             ;CHECK EOL
1E2B  B7                  ORA     A
1E2C  CA E8 18            JZ      ER7
1E2F  3E 02               MVI     A,2             ;CHECK FOR 'TO'
1E31  CD F7 12            CALL    SYMSRT
1E34  FE 07               CPI     7
1E36  C2 86 1F            JNZ     ER17
1E39  23                  INX     H               ;BUMB PNTR'S
1E3A  23                  INX     H
1E3B  79                  MOV     A,C             ;CHECK EOL
1E3C  B7                  ORA     A
1E3D  CA E8 18            JZ      ER7
1E40  CD C7 19            CALL    EVAL            ;EVALUATE LIMIT
1E43  E5                  PUSH    H               ;SAVE H,L
1E44  21 90 21            LXI     H,FLIMT         ;SAVE LIMIT VALUE
1E47  CD A8 17            CALL    COPDH
1E4A  79                  MOV     A,C             ;CHECK EOL
1E4B  B7                  ORA     A
1E4C  C2 5C 1E            JNZ     STP
1E4F  11 05 1E            LXI     D,FONE          ;DEFAULT STEP=1
1E52  E1                  POP     H               ;RESTORE H,L
1E53  C3 6B 1F            JMP     FBILD
1E56  E1          STP:    POP     H               ;GET H,L
1E57  3E 02               MVI     A,2             ;LOOK FOR 'STEP'
1E59  CD F7 12            CALL    SYMSRT
1E5C  FE 08               CPI     8
1E5E  C2 86 1F            JNZ     ER17
1E61  23                  INX     H               ;FIX H,L
1E62  23                  INX     H
1E63  23                  INX     H
1E64  0C                  INR     C               ;CHECK EOL
1E65  CD F6 15            CALL    ICP7
1E68  CD C7 19            CALL    EVAL            ;GET STEP SIZE
                  ;
                  ; AT THIS POINT:
                  ; VARIABLE NAME IS IN LOCATION VNAME
                  ; VARIABLE ADDRESS IS IN LOCATION VLOC
                  ; VARIBLE HAS BEEN INITIALIZED
                  ; LIMIT IS IN 4 BYTE LOCATION FLIMT
                  ; STEP IS POINTED TO BY D,E
                  ; H,L,C ARE POINTER, COUNTER AS USUAL
                  ;
1E6B  D5          FBILD:  PUSH    D               ;SAVE PNTR TO STEP
1E6C  2A 8C 21            LHLD    VNAME           ;GET VARIABLE NAME
1E6F  3E 3F               MVI     A,377Q          ;MASK
1E71  A4                  ANA     H               ;MASK OFF TOP 2 BITS
1E72  67                  MOV     H,A             ;SET UP TO CALL FSYM
1E73  4D                  MOV     C,L
1E74  CD 0D 16            CALL    FSYM            ;FIND ENTRY
1E77  DA 8D 1E            JC      FEXST           ;IT WAS THERE
1E7A  E5                  PUSH    H               ;IT WASN'T, SAVE H,L
1E7B  2A 54 21            LHLD    NXTSP           ;UPDATE NXTSP
1E7E  3E 08               MVI     A,8             ;ADD 8 TO H,L
1E80  85                  ADD     L
1E81  6F                  MOV     L,A
1E82  3E 00               MVI     A,0
1E84  8C                  ADC     H
1E85  67                  MOV     H,A
1E86  22 54 21            SHLD    NXTSP           ;NEW VALUE OF NXTSP
1E89  CD 14 15            CALL    MEMFUL          ;CHECK MEMORY
1E8C  E1                  POP     H               ;GET  ADD. IN DATA BLOCK
1E8D  D1                  POP     D               ;ADDRESS OF STEP SIZE
1E8E  CD A6 17    FEXST:  CALL    COPDH           ;STORE IT
1E91  23                  INX     H               ;PNT TO WHERE VAR. PNTR
1E92  23                  INX     H               ;IS
1E93  23                  INX     H
1E94  23                  INX     H
1E95  3A 8E 21            LDA     VLOC            ;FIRST BYTE
1E98  77                  MOV     M,A             ;STORE IT
1E99  23                  INX     H
1E9A  3A 8F 21            LDA     VLOC+1          ;SECCND BYTE
1E9D  77                  MOV     M,A
1E9E  23                  INX     H               ;PNT TO WHERE LIMIT GOE
1E9F  11 90 21            LXI     D,FLIMT         ;WHERE IT IS NOW
1EA2  CD A6 17            CALL    COPDH           ;COPY IT
1EA5  23                  INX     H               ;PNT TO WHERE KFPNT GOE
1EA6  23                  INX     H               ;S
1EA7  23                  INX     H
1EA8  23                  INX     H
1EA9  3A 56 21            LDA     KFPNT           ;1ST BYTE
1EAC  77                  MOV     M,A
1EAD  23                  INX     H
1EAE  3A 57 21            LDA     KFPNT+1         ;2ND BYTE
1EB1  77                  MOV     M,A
                  ; PUT CURRENT VNAME ON NESTING STACK
1EB2  21 00 00            LXI     H,0             ;GET STACK-POINTER
1EB5  39                  DAD     SP
1EB6  22 8F 21            SHLD    VLOC            ;SAVE IT
1EB9  2A 94 21            LHLD    NEST            ;GET NEST SP
1EBC  7D                  MOV     A,L             ;COMPARE WITH STACK LIM
```

```
1EBD   FE 96              CPI    TOPNS AND 377Q     ;NEED ONLY COMPARE PAGE
1EBF   CA 8B 1F           JZ     ER18               ;FOR'S NESTED TOO DEEPL
1EC2   F9           NSTOK: SPHL                     ;LOAD NEW SP
1EC3   E1                  XCHG                     ;SAVE NEST SP
1EC4   2A 8C 21            LHLD   VNAME            ;GET INDEX NAME
1EC7   E5                  PUSH   H                ;SAVE IT
1EC8   1B                  DCX    D                ;UPDATE NEST SP
1EC9   1B                  DCX    D
1ECA   EB                  XCHG                    ;SAVE IT
1ECB   22 94 21            SHLD   NEST
1ECE   2A 8E 21            LHLD   VLOC             ;RESTURE OLD SP
1ED1   F9                  SPHL
1ED2   C3 8C 1B            JMP    1END             ;ALL DONE
1ED5   80 00 00 00 01 FONE: DB     2000,0,0,0010    ;FLOATING PNT ONE
                       ;
                       ; NEXT STATEMENT PROCESSOR
                       ;
1ED9   2A 58 21      NEXT: LHLD   CPNT             ;FIX PNTR'S
1EDC   23                  INX    H
1EDD   23                  INX    H
1EDE   23                  INX    H
1EDF   0E                  MVI    C
1EE0   CD F6 15            CALL   ICP7
1EE3   CD 22 12            CALL   ALPHA            ;LETTER?
1EE6   D2 9A 1F            JNC    ER21             ;NO, ERROR
1EE9   46                  MOV    B,M              ;YES, GET IT
1EEA   51                  MOV    D,C              ;SAVE C
1EEB   0E 00               MVI    C,0              ;INIT C TO 0
1EED   23                  INX    H                ;BUMP PNTR'S
1EEE   15                  DCR    D
1EEF   CA FD 1E            JZ     NEXT1            ;NUMBER?
1EF2   CD 14 12            CALL   NUMB             ;NO, ERROR
1EF5   D2 9A 1F            JNC    ER21             ;YES, GET IT
1EF8   4F                  MOV    C,M              ;SHOULD BE ECL
1EF9   15                  DCR    D
1EFA   C2 9A 1F            JNZ    ER21
1EFD   21 00 00      NEXT1: LXI    H,0              ;GET SP
1F00   39                  DAD    SP
1F01   22 8C 21            SHLD   VLOC             ;SAVE IT
1F04   2A 94 21            LHLD   NEST             ;GET NEST SP
1F07   7D                  MOV    A,L              ;COMPARE WITH BOTTOM
1F08   FE AA               CPI    BOTNS AND 377Q
1F0A   CA 9D 1F            JZ     ER19             ;NEXT BEFORE FOR
1F0D   F9                  SPHL                    ;LOAD SP
1F0E   E1                  POP    H                ;GET LAST INDEX
1F0F   7B                  MOV    A,B              ;COMPARE TO CURRENT
1F10   BC                  CMP    H
1F11   C2 95 1F            JNZ    ER20             ;NESTING ERROR
1F14   79                  MOV    A,C
1F15   BD                  CMP    L
1F16   C2 95 1F            JNZ    ER20
1F19   2A 8E 21            LHLD   VLOC             ;ALL OK, RESTORE OLD SP
1F1C   F9                  SPHL
1F1D   3E 3F               MVI    A,77Q            ;MASK
1F1F   A0                  ANA    B                ;MASK OUT TOP 2 BITS
1F20   47                  MOV    B,A
1F21   CD 0D 16            CALL   FSYM             ;FIND SYMBOL
1F24   EB                  XCHG                    ;ADDRESS TO D,E
1F25   21 7C 21            LXI    H,FREG1          ;COPY STEP TO FREG1
1F28   CD A8 17            CALL   COPDH
1F2B   13                  INX    D                ;PNT TO CHARACTERISTIC
1F2C   13                  INX    D
1F2D   13                  INX    D
1F2E   1A                  LDAX   D                ;GET IT
1F2F   F6 80               ANI    200Q             ;GET SIGN
1F31   17                  RAL                     ;ROTATE IT INTO CARRY
1F32   3F                  CMC                     ;COMPLEMENT IT
1F33   3E 00               MVI    A,0              ;MAKE SURE A=0
1F35   17                  RAL                     ;ROTATE TO LSB
1F36   3C                  INR    A                ;BUMP BY ONE
1F37   32 8L 21            STA    VLOC             ;SAVE IT, ITS =1 IF - S
1F3A   13                  INX    D                ;PNT TO VARIABLE PNTR
1F3B   EB                  XCHG                    ;GET IT TO DE
1F3C   5E                  MOV    E,M
1F3D   23                  INX    H
1F3E   56                  MOV    D,M
1F3F   23                  INX    H
1F40   E5                  PUSH   H                ;SAVE DATA BLOCK PNTR.
1F41   21 80 21            LXI    H,FREG2          ;COPY VARIBLE VALUE TO
1F44   CD A8 17            CALL   COPDH            ;SAVE VARIABLE LOCATION
1F47   EB                  XCHG
1F48   3E 02               MVI    A,2              ;SET UP TO ADD
1F4A   CD 3B 1A            CALL   BINUP            ;AND DO IT
1F4D   CD A8 17            CALL   COPDH            ;COPY TO VARIABLE
1F50   21 7C 21            LXI    H,FREG1          ;AND TO FREG1 FOR COMPA
1F53   CD A8 17            CALL   COPDH
1F56   D1                  POP    D                ;PNT TO LIMIT
1F57   21 80 21            LXI    H,FREG2          ;COPY TO FREG2
1F5A   CD A8 17            CALL   COPDH
1F5D   D5                  PUSH   D                ;SAVE DATA BLOCK PNTR
1F5E   CD 23 1D            CALL   FCOMP            ;COMPARE
1F61   21 8E 21            LXI    H,VLOC           ;COMPARE WITH STEP TYPE
1F64   BE                  CMP    M
1F65   E1                  POP    H                ;GET DATA BLOCK PNTR.
1F66   CA 74 1F            JZ     NXTDN            ;YES => LOOP DONE
1F69   23                  INX    H                ;LOOP NOT DONE
1F6A   23                  INX    H                ;PNT TO TRANSFER ADD.
1F6B   23                  INX    H
1F6C   23                  INX    H
1F6D   5E                  MOV    E,M              ;GET IT TO H,L
1F6E   23                  INX    H
1F6F   56                  MOV    D,M
1F70   EB                  XCHG
1F71   C3 8C 16            JMP    ILOOP
1F74   21 94 21      NXTDN: LXI    H,NEST           ;POP NEST STACK
1F77   34                  INR    M
1F78   34                  INR    M
1F79   C3 8C 1B            JMP    1END             ;CONTINUE
1F7C   3E 09         ER9:   MVI    A,9
1F7E   C3 BE 15            JMP    ERROR
1F81   3E 16         ER16:  MVI    A,16H            ;'=' EXPECTED(NOTE: NO
1F83   C3 BE 15            JMP    ERROR            ;FOR INDICES)
1F86   3E 17         ER17:  MVI    A,17H            ;BAD SYNTAX NEAR 'TO' O
1F88   C3 BE 15            JMP    ERROR            ;IN FOR STATEMENT
1F8B   3E 18         ER18:  MVI    A,18H            ;FOR'S NESTED TOO DEEPL
1F8D   C3 BE 15            JMP    ERROR
1F90   3E 19         ER19:  MVI    A,19H            ;'NEXT' EXECUTED BEFORE
1F92   C3 BE 15            JMP    ERROR            ;FOR
1F95   3E 20         ER20:  MVI    A,20H            ;NESTING ERROR, 'FOR'-'
1F97   C3 BE 15            JMP    ERROR
1F9A   3E 21         ER21:  MVI    A,21H            ;BAD INDEX IN FOR-NEXT
1F9C   C3 BE 15            JMP    ERROR
NO PROGRAM ERRORS
                            END
```

SYMBOL TABLE

```
* 01
A       0007    ADDJ    1A56    ADFLD   1812    AFOND   190F
AGA     1A02    ALPHA   1222    AM      1614    ARCK    1B89
ARYES   18CD    AS8C    1A59    B       0000    BAC     1220
BINOP   1A3B    BMPTR   19BB    BND1    156C *  BND2    15A3
BND3    1546    BUTNS   21AA    BOUND   1540    BUPT    1893 *
C       0001    C1      1219    CALLP   1058    CHAR2   0FF7
CHAR5   1508    CHEKE   1981    CHK1    12A3    CHKMD   1987
CLSUB   10CC    CONT    177B    CONV    0FE5    CUPD1   17AE
COPDH   17A8    CPNT    2158    CREG    2184    CVB     122A
CVB1    124B    CVB2    1268    D       0002    DCOMP   0FF1
DFXL    0FDC    DIM     1720    DIV     1A8D    DLOOP   1726
DOL     19BC    E       0003    ECAV    19D1    ENDD    16F2
ENTRY   167F    EOK     19FA *  ER10    1C78    ER14    1EA3
ER16    1F81    ER17    1F86    ER18    1F8B    ER19    1F90
ER20    1F95    ER21    1F9A    ER6     172C    ER7     18E8
ER8     1A22    ER9     1F7C    ERLN    15DE    ERRET   1BC8
ERROR   15BE    ERRK1   15D2    EVAL    19C7    EXPRE   1AD6
F1      144E    FALSE   1D12    FBAC    1676    FBILD   1E6B
FCUMP   1023    FDAT    196F    FEXST   1E8D    FINPT   0FE8
FIX     1915    FLDFU   182D    FLIMT   2190    FMULT   1A87
FNDSB   10BE    FONE    1ED5    FOR     10E7    FORIQ   1441
FORI1   1440    FORL2   143F    FORM1   1448    FORM2   1447
FORM3   1446    FORM4   1445    FORM5   1449    FORM6   144F
FORM7   1444    FORM8   1443    FORM9   1442    FOUND   12A0
FPERR   1A5B    FREG1   217C    FREG2   2180    FSYM    160D
F=M     21AA    GET     1709    GKEY    16C6    GOOD    1932
GOSUB   1D0D    GOTFI   1700    GREG    2177    GSENT   1703
GTRA    1709    H       0004    HLINP   2186    HOME    1824
HSRIN   1405    IAGA    1C8D    IBUF    2101    ICP2    1605
ICP4    1630    ICP7    15F6    ICP8    15FB    IDONE   1983
IEND    1B8C    IFRT    1C7D    III     1CA8    ILOOP   16BC
INCPT   1607    INP     1973    INPER   1853    INPOK   1B89
INPUT   184C    INSER   108F    ISRI0   116B    ISRI1   1171
ISRI2   117A    ISRIA   10A2    ISRI7   10BD    ISRT2   10C4
ISRT3   1304    ISRT4   10D7    ISRT5   110F    ISRT6   1121
ISRT7   1144    ISRT8   1148    ISRT9   1152    JTBL    16D6
KASE    2160    KDAT1   134F    KDAT2   1364    KDAT3   139E
KDAT4   13C0    KDATA   1348    KFPNT   2156    KL2     1154
KL4     2156    KL6     2158    KLEN    2158    KLINE   2152
KONT    1B3B    L       0005    L1      15AE    L2      1288
LADD    0FD0    LDIV    0FD6    LEN     2161    LENGT   1271
LET     18DD    LIS1    11EA    LIST    11D1    LMCM    0FDF
LMUL    0FD3    LOK     18ED    LPNT    2152    LSUB    0FD9
LUKON   1627    M       0006    M1      1000    M1A     1013
M2      1016    M3      101F    M4      1042    M4A     1047
MCHK    0FF4    MDBC    1A90    MEMEN   27FF    MEMFU   1514
MEMSI   2100    MESCR   2188    MINSE   1930    MODE    2185
MODE1   1997    MORE    187F    MULT    0FE8    MULT1   2162
MULT2   2164    NDOU    175F    NEST    2194    NEXT    1ED9
NEXT1   1EF0    NFPER   17AE    NL2     2140    NL4     214F *
NL6     2151    NLE1    1275    NLE2    1284    NLINE   2148
NOENT   1646    NOLIN   1191    NOMAT   1638    NOSYM   1644
NOT3    1D08    NSRCH   1288    NSTOK   1EC2 *  NUMB    1214
NUSUB   1D7D    NXTON   1F74    NXTSP   2159    OBUFF   2100
ODA10   14CB    ODAT1   14E4    ODAT2   1464    ODAT3   1478
ODAT4   1481    ODAT5   148B    ODAT6   1491    ODAT7   14A0
ODAT8   1487    ODAT9   14C2    ODATA   145A    OKK     1847
OKLET   1731    OKN     1715    ONWD    1825 *  OUTR    1788
P1      1288    P2      12C2    PAD     12AD    PARLP   1D9D
PEND    1846    PIN1    14DB    PIN1A   14DE    PIN3    1500
PINST   1C73    PK1     1418    PK2     1435    PL2     2158
PL4     215D    PL6     215F    PLINE   2159    PLOOP   1AAE
PREXP   1DB5    PR1     1A95    PR11    1C63    PRMPT   185C
PRTIT   1AED    PSW     0006    PTI     106D    PTAPE   1065
PTFIN   1C4C    PTVAL   0FEE    QCHEK   14C8    QOTOK   1ABD
QSTR1   1430    QSTRG   142B    QTCHK   120E *  QUITT   1208
QUOTE   1AB4    RDKON   1853    RELAT   1CD5    RETRN   1DE5
RINST   1B36    RUN     1678    S2      1307 *  S3      1308
S4      1323    S4A     1322    S5      133F    SAVV    1C32
SBSAV   215D    SC1     1876    SCALR   18B1    SCMEM   21AD
SCOLN   1B07    SCR     2166    SEND    0006    SFSC    1894
SINEQ   1D3B    SLOAD   18AA    SONWD   1B39    SORCE   169A
SP      0006    SPACE   1981    STAC    2196    STLIN   2149
STNER   18CB    STOKK   18AF    STOKV   18A0    STP     1E56
STPNT   11BB    STRIN   1B92    STSIZ   0014    STSPA   2148
SUBB    1A81    SUBS    21AC    SYMSR   12F7    TAPE    11AA
THEN    1D19    TIN1    13CC    TIN2    13F6    TIN4    1408
TIN4A   1408    TIN5    1401    TIN5A   1404    TOPNS   2196
TRUE    1A76    TTYIN   13C9    V1      1815    VALUE   17C3
VAR     186D    VARAD   218A    VEND    21AA    VLOC    218E
VNAME   218C    W1      12DF    W2      12F0    WFOR    1427
WFPER   1A77    WHAT    119E    WRIT    12D0    WRIT1   12D2
ZROL    0FCD
```


INTERFACE AGE IS SEEKING ARTICLES

Here is the opportunity for Computer Clubs to benefit. For each article accepted from a club member and containing an additional cover letter signed by the club's Secretary, INTERFACE AGE will pay a **$25.00 bonus** to the club's treasury over and above the customary author's honorarium and the club will be included in the by line.

ATTENTION All Ye Alice Freaks and Other Lovers of Logical Systems. Here is documented evidence that Lewis Carroll would have read BYTE had he lived in 1976:

JABBERWOCKY.

'Twas brillig, and the slithy toves
  Did gyre and gimble in the wabe;
All mimsy were the borogoves,
  And the mome raths outgrabe.

"Beware the Jabberwock, my son!
  The jaws that BYTE, the claws that catch!
Beware the Jubjub bird, and shun
  The frumious Bandersnatch!"

He took his vorpal sword in hand:
  Long time the manxome foe he sought—
So rested he by the Tumtum tree,
  And stood awhile in thought.

And as in uffish thought he stood,
  The Jabberwock, with eyes of flame,
Came whiffling through the tulgey wood,
  And burbled as it came!

One, two! One, two! And through and through
  The vorpal blade went snicker-snack!
He left it dead, and with its head
  He went galumphing back.

"And hast thou slain the Jabberwock?
  Come to my arms, my beamish boy!
O frabjous day! Callooh! Callay!"
  He chortled in his joy.

'Twas brillig, and the slithy toves
  Did gyre and gimble in the wabe;
All mimsy were the borogoves,
  And the mome raths outgrabe.

Don't miss out on all the fun and high quality information which is found in every issue of BYTE. Subscribe today. Join the tea party and have a BYTE to eat.

Name .....
Address .....
City ..... State..... Zip.....
☐ BILL ME ☐ Check for $12 enclosed
☐ Bill BankAmericard or MasterCharge # .....

Name .....
Address .....
City ..... State..... Zip.....
☐ BILL ME ☐ Check for $12 enclosed
☐ Bill BankAmericard or MasterCharge # .....

• BYTE • 70 Main St • Peterborough NH 03458

Allow six weeks for processing.

CIRCLE INQUIRY NO. 46

An Inside Look Into NIBL – Extended Tiny BASIC for the SC/MP

By Mark Alexander

National Semiconductor Corp.
Santa Clara, CA

INTRODUCTION

NIBL (National Industrial Basic Language) is a conversational programming language for the SC/MP. It is a language similar to Tiny BASIC Extended, but it also has some unique features. Many of these features, such as a genuinely useful control structure (the PASCAL-influenced DO/UNTIL) and the indirect operator ("@") have been added to the language to allow NIBL to be nearly as flexible as machine language in such applications as medium-speed process control.

By using NIBL, one trades the high execution speed and low memory consumption of machine language for some very tangible advantages: program readability and modifiability which are truly difficult to achieve in machine language programs.

NIBL programs are interpreted by a large (4K byte) TBX SC/MP program that resides in ROM. The interpreter is broken into two main blocks: a program written in an Intermediate (or Interpretive) Language — IL for short — which does the actual interpretation; and a collection of SC/MP machine language subroutines invoked by the IL program. The IL approach is well-documented in vol. 1, no. 1 of ***Dr. Dobb's Journal of Computer Calisthentics and Orthodontia***, and readers should refer to that issue for a more detailed description of the interpretation process.

In Table 1, the formal grammar for NIBL is given. This is the ultimate authority (other than the interpreter itself) on how legal NIBL statements are formed. The following descriptions of the NIBL statements will refer to portions of the grammar. Table 2 contains a list of the error messages produced by the NIBL system. Finally, a listing of the NIBL interpreter is given.

Table 1: NIBL grammar

On reading the grammar:

All items in single quotes are actual symbols in NIBL; all other identifiers are symbols in the grammar. The equals sign "=", means "is defined as"; parentheses are used to group several items together as one item; the exclamation point, "!", means an exclusive or choice between the items on either side of it; the asterisk, "*", means zero or more occurrences of the item to its left; the plus sign, "+", means one or repetitions; the question mark, "?", means zero or one occurrences; and the semicolon, ";", marks the end of a definition.

```
NIBL Line          =  Immediate Statement
                   !  Program Line
                   ;

Immediate Statement = (Command ! Statement) Carriage Return;

Program Line = (Decimal Number Statement List Carriage Return);

Command            =  'NEW'
                   !  'CLEAR'
                   !  'LIST' Decimal Number ?
                   !  'RUN'
                   ;

Statement List = Statement (':' STATEMENT) *;

Statement          =  'LET' ? Left part '=' Rel Exp
                   !  'LET' ? '$' Factor '=' (String ! '$' Factor)
                   !  'GO' ('TO' ! 'SUB') Rel Exp
                   !  'RETURN'
                   !  ('PR' ! 'PRINT') Print List
                   !  'IF' Rel-Expr 'THEN' ? Statement
                   !  'DO'
                   !  'UNTIL' Rel-Exp
                   !  'FOR' Variable '=' Rel Exp 'TO' Rel Exp ('STEP'
                      Rel Exp) ?
                   !  'NEXT' Variable
                   !  'INPUT' (Variable + ! '$' Factor)
                   !  'LINK' Rel Exp
                   !  'REM' Any Character Except Carriage Return +
                   !  'END'
                   ;

Left Part = (Variable ! '@' Factor ! 'STAT' ! 'PAGE') ;

Rel Exp            =  Expression Relop Expression
                   !  Expression
                   ;

Relop = '<' ! '<' '=' ! '<' '>' ! '>' ! '>' '=' ! '=' ;

Expression         =  Expression Adding Operator term
                   !  ('+' ! '−') ? Term
                   ;

Adding Operator = '+' ! '−' ! 'OR' ;

Term               =  Term Multiplying Operator Factor
                   !  Factor
                   ;

Multiplying Operator = '*' ! '/' ! 'AND' ;

Factor             =  Variable
                   !  Decimal Number
                   !  '(' Rel Exp ')'
                   !  '@' Factor
                   !  '#' Hex Number
                   !  'NOT' Factor
```

```
                    ! 'MOD' '(' Rel Exp ',' Rel Exp ')'
                    ! 'RND' '(' Rel Exp ',' Rel Exp ')'
                    ! 'STAT'
                    ! 'TOP'
                    ! 'PAGE'
                    ;

Variable = 'A' ! 'B' ! 'C' ! . . . ! 'Y' ! 'Z' ;

Decimal Number = Decimal Digit + ;

Decimal Digit = '0' ! '1' ! '2' ! . . . ! '9' ;

Hex-Number = (Decimal Digit ! Hex Digit) + ;

Hex Digit = 'A' ! 'B' ! 'C' ! 'D' ! 'E' ! 'F' ;

Print List = Print Item + ;

Print Item = (String ! Rel Exp ! '$' Factor) ;

String = '"' Almost Any Character '"' ;
```

NOTE: Spaces are not usually significant in NIBL programs, with the following exceptions: spaces cannot appear within key words (such as 'THEN' or 'UNTIL') or within constants. Also, a variable (such as A or Z) must be followed immediately by a non-alphabetic character to distinguish it from a key word.

Table 2: NIBL error messages

Error messages are of the form:

EEEE ERROR AT LN

where EEEE is one of the error codes below, and LN is the number of the line in which the error was encountered.

AREA	No more room left for program in current page
CHAR	Character after logical end of statement
DIVØ	Division by zero
END"	No ending quote on string
FOR	FOR without NEXT
NEST	Nesting limit exceeded in expression, FOR's, GOSUBs, etc.
NEXT	NEXT without FOR
NOGO	No line number corresponding to GOTO or GOSUB
RTRN	RETURN without previous GOSUB
SNTX	Syntax error
STMT	Statement type used improperly
UNTL	UNTIL without DO
VALU	Constant format or value error

HISTORY OF NIBL

NIBL came into this world as an interpreter for Tiny BASIC, as originally described in the first issue of Dr. Dobb's Journal. That program was written by Steve Leininger, who subsequently left before the program was ever assembled or executed. The current version of NIBL is an almost complete re-write of the original interpreter, with changes and additions being made to improve the modularity of the program, to greatly increase execution speed, and to extend the capabilities of the language itself.

SYSTEM REQUIREMENTS

The NIBL interpreter is intended to be a ROM-resident program in the first 4K of the SC/MP Memory address space (although it will run just as well in RAM). The interpreter requires at least 2K bytes of RAM starting at address 1000 (base 16), of which the interpreter uses nearly 300 bytes for stacks, variables, etc., leaving the rest for the user's program. Another 2K bytes of memory may be added to fill up this 4K page, forming what is hereafter referred to as "Page 1."

The SC/MP architecture forces memory to be split into pages of 4K bytes each; therefore, NIBL allows seven such pages to be used for storing programs. NIBL programs in the seven pages are edited separately, but may be linked together during program execution by special NIBL statements described in the following pages. Page #1 as mentioned above, must be RAM since the interpreter uses part of it as temporary storage; the part used to store application programs starts at location 111E (base 16).

The other six pages of memory, each of which starts at location n000 (base 16), where n is the page number, may be either RAM or ROM. Page 2 is a special page: it can contain a NIBL program to be executed immediately upon powering up the NIBL system.

The memory organization of NIBL is shown in Figure 1.

Throughout this article, the assumption is made that the user has a teletype with paper tape reader and punch, as with the SC/MP Low Cost Development System (LCDS). In fact, NIBL was designed to use the SC/MP LCDS teletype interface, but to be completely independent of the LCDS firmware. If NIBL is to be run on its own, the system should have the same configuration for the teletype, with the reader relay being operated directly by the SC/MP. At present, paper tape is the only medium for saving NIBL programs, but as soon as the hardware and software for a SC/MP cassette interface become available, NIBL will be able to link to routines for saving and loading programs with ease.

Since the teletype interface is not based on a UART, the terminal baud rate can only be changed by modifying the timed delays in NIBL's I/O routines. NIBL has been run successfully at 1200 baud with a CRT terminal; the assembly listing of the NIBL program is programmed for a 110 baud rate system.

COMMUNICATING WITH NIBL

When the NIBL system is ready to accept input, it prompts at the teletype with a ">" prompt sign. (NIBL is now in "edit mode.") The user then enters a line terminated by a carriage return. There are several special characters that are used to edit lines as they are typed:

Shift/O (back-arrow) causes the last character typed to be deleted.

Control/H (backspace) performs the same function as shift/O, but echoes as a backspace/space/backspace sequence, which is only useful if the teletype routines are modified to run at high baud rates for use with CRT's.

Control/U (echoed as " ' U") causes the entire line to be deleted; NIBL reprompts for a new line.

Entering a line to NIBL without a leading line number causes the line to be executed directly by NIBL. Most NIBL statements, as well as the four program control commands, may be executed in this manner.

A line with a leading number (in the range 0 through 32767) is entered into the NIBL program in the cur-

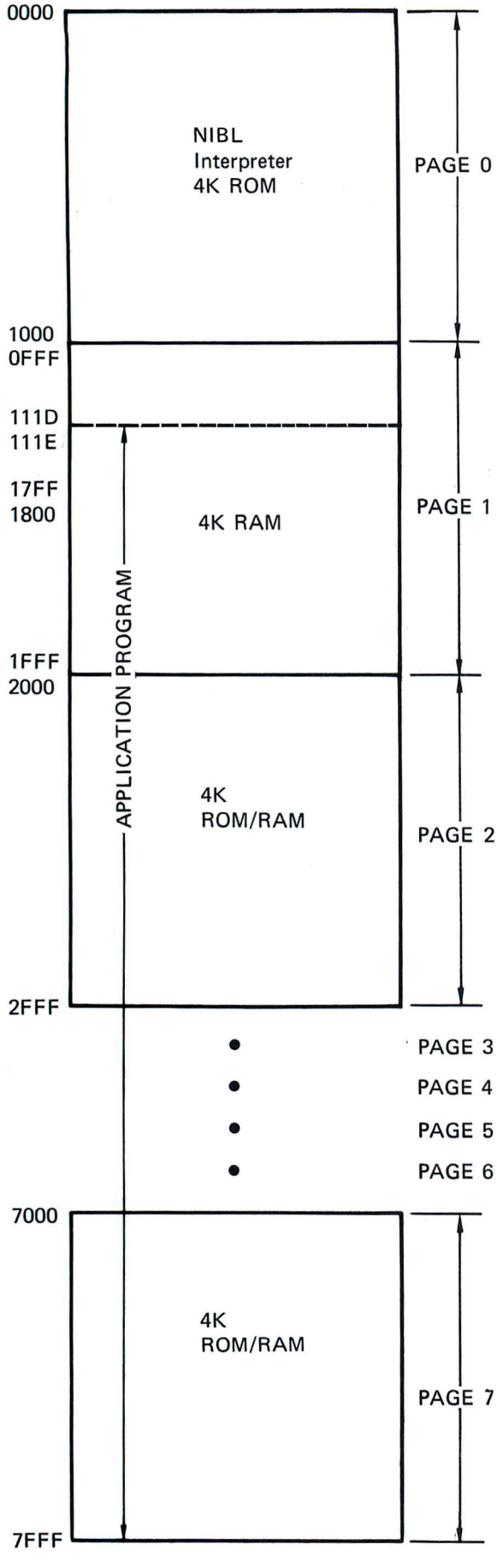


FIGURE 1. NIBL MEMORY ORGANIZATION

rent page. (Make sure that the value of the pseudo-variable PAGE is valid, so that the line isn't lost into non-existent memory.) The NIBL editor sorts the program lines as they are entered into ascending order by line number.

Typing a line number followed by a carriage return deletes that line from the program. Typing a line with the same number as an existing line's causes the new line to replace the old one in the program.

Each of the seven memory pages may contain a different program, separate from the rest. Editing the program in one page will not affect the other pages. To switch editing from one page to another, simply type PAGE = n, where n is the number of the new page.

VARIABLES

There are twenty-six variable names in NIBL: the letters A through Z. They are all 16-bit binary variables, so they can be used to hold addresses as well as signed numeric data. Space for pre-declared variables is allocated for them in RAM when NIBL powers up.

CONSTANTS

NIBL allows either decimal or hexadecimal (base 16) constants to appear in expressions. Decimal constants must lie in the range 0 through 32767; the unary minus ("−") is used to obtain negative values. The value −32768 is a valid NIBL integer, but it is not legal as it stands. To represent it, use −32767-1 or #8000 instead.

Hexadecimal constants are denoted by a pound sign ("#") followed by a string of hexadecimal digits (0-9, A-F). NIBL does not check for overrun in hex constants; consequently, only the 4 least significant digits of the hex digit string are kept.

FUNCTIONS

NIBL includes three functions that may appear in any expression. These are described as follows:

RND (X, Y) returns a pseudo-random integer in the range X through Y, inclusive, where X and Y are arbitrary expressions. In order for the function to work properly, the value of Y − X should be positive and no greater than 32767.

MOD (X, Y) returns the absolute value of the remainder from X divided by Y (where X and Y are expressions).

TOP (with no arguments) returns the address of the first free byte in the memory page currently being edited or executed. In other words, it is the address of the last byte at the top of the current NIBL application program in the current page, plus one.

PSEUDO-VARIABLES

NIBL has two pseudo-variables in addition to the standard variables. These are STAT and PAGE. Both of these variables may appear on either side of an assignment statement.

STAT represents the SC/MP status register. The current value of the status register can be referred to by using STAT in an expression; or an assignment may

be made to the status register by executing a statement such as STAT = 4 or STAT = STAT OR #20. When NIBL makes an assignment to the status register in this manner, it clears the interrupt-enable bit of the value before it is actually assigned. Note also that only the lower byte of the value is assigned, the high byte is ignored.

The carry and overflow bits in STAT are meaningless since the NIBL system is continually modifying them. The utility of STAT lies in the fact that 5 of its bits are connected to I/O sense lines on the SC/MP chip.

The pseudo-variable PAGE contains the number of the memory page currently being executed or edited. As indicated in Figure 1, there are seven pages in which NIBL programs may be stored; therefore, the PAGE number may be only in the range of 1 through 7. If an assignment of a value outside this range is made to PAGE, only the 3 least significant bits LSBs of the value are used. Page one is defined as LSBs = 000, page two as LSBs = 001 etc.

If PAGE is modified while NIBL is in edit mode, all subsequent editing will take place in the new page.

If PAGE is modified by a NIBL program during execution, control will be passed to the first line of the NIBL program in the new page. This transfer would be effected by a statement such as PAGE = 6 or PAGE = PAGE + 1. Thus, several NIBL programs residing in different 4K pages may be linked together as one large program, if need be. This would allow one to write a 28K STAR TREK program in NIBL, a Herculean and indeed foolish task.

Control may also be transferred from one page to another by three other statements: RETURN, NEXT, and UNTIL. Thus, the first part of a subroutine or loop may be in one page, and the second part may be in another (with control being transferred between the two parts by an assignment to PAGE). In these three special cases, NIBL automatically updates the value of PAGE as the statements are executed.

RELATIONAL OPERATORS

NIBL provides the standard BASIC relational operators, for comparing the values of integer expressions. The operators are as follows:

= equal to
<= less than or equal to
>= greater than or equal to
<> not equal to
< less than
> greater than

All of these operators produce 1 as a result if the relation is true, and 0 if the relation is false. Note that the relational operators may appear anywhere that an expression is called for in the NIBL grammar, not only in IF statements.

ARITHMETIC OPERATORS

NIBL provides the four standard arithmetic functions: addition (+), subtraction or unary minus (−), multiplication (*), and division (/). Since only integers are allowed in NIBL, all quotients are truncated (the MOD function can be used to obtain remainders from division). Any overflow or underflow (other than division by zero) is ignored by NIBL; the reasoning behind this is that it may often be necessary to treat NIBL expressions as unsigned values, such as when performing calculations using memory addresses as the operands. Thus the value of 32767 + 1 is −32768 (or in hexadecimal, °7FFF + 1 = #8000, which makes more sense).

LOGICAL OPERATORS

In NIBL, there are three logical operations that may be performed on values: AND, OR, and NOT. The first two are binary operators, and the latter is unary. All three perform bitwise logical operations on 16-bit arguments, producing 16-bit results. AND, OR, and NOT are sufficient to simulate any other logical operation, through various combinations of the operators.

THE INDIRECT OPERATOR

The indirect operator "@" realizes the functions of PEEK and POKE operations in other BASICs, but with somewhat more elegance. The "@" sign followed by an address (which can be a constant, variable, or expression in parentheses) denotes the contents of that address in memory. Thus, if memory location 245 (decimal) contains 60, the statement X=@245 would result in the value 60 being assiged to X. The indirect operator may also appear on the left side of an assignment statement. For example, @X=@(Y+10) would result in the memory location pointed to by X being assigned the value of the memory location pointed to by the value Y+10.

Use of the indirect operator is not limited to reading from or writing to memory: it also provides a simple way to communicate with peripheral devices that are interfaced to the SC/MP through memory addresses. Note that the "@" operator can only access memory one byte at a time, and that when an assignment is made to a memory location, only the low order byte of the value is moved to the location; the high order byte is ignored.

The indirect operator can also be used to simulate arrays in NIBL. For example, if we wish to define an M x N matrix of one-byte positive integers, we can access the (I,J)th element of the matrix (assuming that (0, 0) is a legal element in the matrix) with the expression @(A+I*N+J). An assignment could be made to that same element by placing the expression on the left side of an assignment statement.

EXPRESSIONS

Expressions in NIBL are composed of variables, constants, function references, pseudo-variables, and operators. Operators bind these other elements together to form complete expressions. NIBL expressions are all 16-bit integers. Evaluation of expressions takes place left-to-right, and the order in which operations take place is determined by operator precedence and the presence of parentheses. The order of evaluation can be deduced from the grammar in Table 1; here is a table of operator precedence:

Lowest precedence (applied last): <, >, <=, >=, =, <>
+, −, OR
*, /, AND

Highest precedence (applied first): @, NOT

Despite this, it is still safest to use plenty of parentheses in expressions to make the intent clear.

PROGRAM CONTROL COMMANDS

LIST causes the entire program in the current page to be listed. Listing can be halted by hitting any key on the teletype: the BREAK key works best.

LIST <number> causes listing to begin at the given line number (or the nearest one greater than the number), rather than at the first line.

LISTing a program is the method used to save it on paper tape. To accomplish this, type LIST with the punch off, then turn on the punch and hit carriage return. After the program is dumped, type a Shift/O with teletype on LOCAL so that the last character (a ">") will be deleted when the tape is entered to NIBL at a later time. NIBL will accept a tape made in this fashion at any time during edit mode. The tape reader is enabled at all times by NIBL, and it does not distinguish between the reader and the keyboard when accepting input. Superfluous line-feed and null characters on the tape are echoed but ignored.

RUN causes three actions: first, all variables are zeroed; secondly, all stacks (the FOR, DO, and GOSUB stacks) are cleared; and finally the program in the current page is executed, starting with the first line in sequence.

RUN is not the only way to start program execution: GOTO and GOSUB can also be used to jump into a program from edit mode. For example, if there is a subroutine at line 1000 that is being tested, typing GOSUB 1000 will cause that routine to be executed, with NIBL returning to edit mode upon encountering a RETURN statement. When GOTO and GOSUB are used to run a program, the variables and stacks are not cleared.

Hitting any key while a program is being run will cause NIBL to break execution, printing a message and the line number where the break was detected. The BREAK key on the teletype works best for this.

CLEAR causes all variables to be zeroed and the three stacks mentioned above to be cleared. This latter feature of the CLEAR command is quite useful after a stack nesting error has occurred (for example, if GOSUBs are nested more than eight levels deep).

NEW clears the program in Page 1, and changes the value of PAGE to 1. This is the form of the command most likely to be used by NIBL novices who do not wish to be confused by the page selection features of NIBL. NEW should be the first thing one types in to NIBL when first powering up.

NEW <number> sets the value of PAGE to the <number>, and clears the program in that page.

ASSIGNMENT STATEMENTS

Previously, two different types of assignment statements have been described: assignments to the pseudo-variables STAT and PAGE, and assignments to memory locations with the indirect operator. Another form of the assignment statement is the conventional assignment to a variable (A − Z), e.g. A = A + 1 or A = 32 < (4 * I). There are also statements which look like string assignments, but these are not standard BASIC, and are described later in the section on string handling. The word "LET" is optional in front of any assignment statement (leaving it out increases execution speed, unlike most Tiny BASIC systems).

IF/THEN STATEMENT

The IF statement allows conditional execution of one or more statements (as many as can fit on one line). The syntax for the IF statement is:

'IF' Rel-exp 'THEN'? Statement

which indicates that the word THEN is optional, and that any statement (including another IF statement) may follow the conditional expression. If the IF condition is true (i.e. is non-zero), the statement following it (and any others on the line) will be executed; otherwise, control immediately transfers to the next program line. The condition does not need to contain relational operators: a statement such as IF MOD (A, 5) THEN . . . is perfectly legal. In this example, the statement following the THEN would be executed if A were not divisible by 5.

GOTO, GOSUB, AND RETURN STATEMENTS

The syntax for the GOTO statement is 'GOTO' followed by an expression. The effect of the GOTO statement is to transfer control to the line whose number is indicated by the expression. An error occurs if the specified line does not exist in the current page. Unlike standard BASICs, any arbitrary expression can be used to specify the line number, as well as the usual decimal constant. This allows computed branches to be performed with the same effect as the ON . . . GOTO statement in standard BASIC.

The GOSUB statement is identical to the GOTO statement in form. It too causes a branch to a new line, but it also saves the address of the following statement on a stack. When a RETURN statement is executed, the saved address is popped from the stack, and control returns to that point in the program. Since an actual address, not a line number, is saved on the GOSUB stack, GOSUB statements may appear anywhere on a multiple-statement line.

GOSUBs may be nested up to eight levels deep; an error will occur if an attempt is made to exceed this limit. The error condition does not destroy the previous contents of the stack, so a RETURN statement can be executed (even in edit mode) without an error occurring. However, any modification of the NIBL program will clear the GOSUB stack, so that a subsequent RETURN without a GOSUB will cause an error.

DO AND UNTIL STATEMENTS

The DO and UNTIL statements are useful in writing program loops efficiently, without using misleading

GOTO statements. Enclosing a group of zero or more statements between a DO statement and an UNTIL <condition> statement (where <condition> is an arbitrary expression) will cause the statement group to be repeated one or more times until the <condition> becomes true (i.e. non-zero). As an example of the use of the DO and UNTIL statements, we present a program that prints the prime numbers:

```
 10 PRINT 1: PRINT 2
 20 I=3
 30 DO
 40 J=I/2: N=2
 50 DO
 60 N=N+2
 70 UNTIL (MOD (I, N) =0) OR (N>J)
 80 IF N>J PRINT I
 90 I=I+2
100 UNTIL 0
```

DO loops may be nested up to eight levels deep, and NIBL acts in the same manner if an overflow occurs as it does with a GOSUB overflow. NIBL also reports an error if an UNTIL statement occurs without a previous DO. A single DO loop may have more than one UNTIL statement as a terminator. For example, if one wished to exit abnormally out of a DO loop and transfer to some appropriate line, it could be done in the following manner:

```
UNTIL 1: GOTO X
```

where X is the line number.

Neither the DO nor the UNTIL statement may be executed in edit mode.

FOR AND NEXT STATEMENTS

The NIBL FOR statement is virtually identical to that in standard BASICs; consequently, it is not explained in great detail here.

As in most BASICs, both positive and negative STEPs are allowed in the FOR statement, and a STEP of +1 is assumed if the STEP portion of the statement is omitted. A FOR loop is terminated by a NEXT <variable> statement, and the <variable> must be the same as that referred to in the FOR statement at the beginning of the loop.

FOR loops may be nested four levels deep; NIBL reports an error if this limit is exceeded, or if a NEXT statement occurs without a previous FOR statement. As with the DO and UNTIL statements, FOR and NEXT may not be executed in edit mode.

Perhaps the only differences between the NIBL FOR statement and that of more elaborate BASICs (such as DEC's BASIC-PLUS for the PDP-11) are that a FOR loop is always executed at least once, and that when a NEXT statement is executed, the STEP value is added to the variable before the test is made to determine if the loop should be repeated (rather than after the test).

INPUT STATEMENT

There are two types of INPUT statements in NIBL: numeric input and string input. The form of the first type is 'INPUT' followed by a list of one or more variables. When this statement is executed, NIBL prompts at the teletype with a question mark ("?"). The user responds with a list of expressions separated by commas, and terminated by a carriage return. For example, a legal response to the statement INPUT A,B,C would be #3FA,26,4*27. These three expressions would then be assigned to the variables A,B, and C, respectively. An illegal response (too few arguments or improper expressions) will result in a syntax error. Any extra arguments in the response are ignored.

The second type of INPUT statement allows strings to be input. The form of the statement is 'INPUT' '$' <address>, where <address> is a Factor, syntactically (usually a variable, constant, or expression in parentheses). When this statement is executed, NIBL prompts the user as before, at which point the user enters a line terminated by the usual carriage return. NIBL then stores the line in memory in consecutive locations, beginning at the address specified. Thus, INPUTS #6000 would cause the input line to be stored starting at location 6000 (base 16); the carriage return would also be stored at the end of the line.

Strings input in this manner can be tested and manipulated by using the "@" operator or the string handling statements described below. They can also be displayed by a PRINT statement.

Neither of the two INPUT statements may be executed in edit mode.

PRINT STATEMENT

The form of the PRINT statement is 'PRINT' or 'PR' followed by a list of print items separated by commas, and optionally terminated by a semicolon, which suppresses an otherwise automatic carriage return after all items in the list are printed.

A print item consists of one of the following:

- A quoted string, which is printed exactly as it appears (with the quotes removed)
- An expression, which is evaluated and printed in decimal format, with either a leading space or a minus sign ("−"), and one trailing space
- A reference to a string in memory, denoted by '$' <address>, where <address> is a Factor as usual. Successive memory locations, starting at the specified address, are printed as ASCII characters, until a carriage return (which is not printed) is encountered.

There is no zone spacing in the PRINT statement, nor does NIBL perform an automatic carriage return/line feed after printing 72 characters. NIBL is not an output-oriented language; fancy formatting has been sacrificed for more useful control structures and data manipulation features. (A subroutine to print a number and skip to the next print zone is a trivial to write in NIBL — it takes about two lines of code, with the DO/UNTIL and FOR/NEXT.)

STRING HANDLING STATEMENTS

String handling in NIBL is very minimal and low-level. The string handling features of the INPUT and PRINT statements have already been mentioned; NIBL provides two more statements for manipulating strings.

A statement such as $<address> = "THIS IS A STRING" would cause the quoted string to be stored in

memory starting at the specified address (which again is a Factor), with a carriage return being appended to the string.

Another statement allows the programmer to move strings around in memory once they have been created. The form of this statement is '$' <destination> '=' '$' <source>, where both <destination> and <source> are Factors, and are the addresses of strings in memory. This statement causes all the characters in the string pointed to by <source> to be copied one-by-one to the memory pointed to by <destination>, until a carriage return (also copied) is encountered. Overlapping the source and destination addresses can produce disastrous results, such as wiping out the entire contents of the current page. Consequently, a string move can be aborted by hitting the BREAK key on the teletype (but it must be done quickly!).

Note that all strings referred to in these statements, and in the INPUT and PRINT statements, are assumed to lie within a 4K page, and wraparound is a possibility which must be anticipated by the programmer. (Long-time SC/MP programmers will be familiar with this minor problem.)

Using these statements, it should be very easy to develop a set of NIBL subroutines for performing concatenation, comparison, and substring operations on strings.

END STATEMENT

The END statement may appear anywhere in a NIBL program and not necessarily at the end. It causes a message and the current line number to be printed, with NIBL returning to edit mode. The END statement is useful when debugging programs, since it acts as a breakpoint in the program that can be removed easily.

LINK STATEMENT

The LINK statement allows NIBL programs to call SC/MP machine language routines at any address. A statement of the form 'LINK' <address>, where <address> is an arbitrary expression, will cause the NIBL system to call the routine at that address by executing an appropriate XPPC P3 (Exchange contents of the Program Counter with designated Pointer Register P3) instruction. The user's routine should make sure that it returns by executing another XPPC P3, and that the value of P3 (Pointer Register #3) upon entry to the routine is restored before returning. The routine may make use of the fact that P2 is set by NIBL to point to the beginning of the RAM block used to store the variables A through Z, with each variable being stored low byte first, high byte second. Thus, parameters may be passed between NIBL programs and machine language routines through the variables. Both P1 and P2 may be modified by the user's routines; they are automatically restored by the NIBL system upon return. The user should be careful not to modify RAM locations with negative displacements relative to P2, or the locations with displacements greater than 51 relative to P2. These locations are used by the interpreter.

REMARK STATEMENT

A comment can be inserted into a NIBL program by preceding it with the word REM. REM causes the rest of the line to be ignored by NIBL during execution. Remarks are useful in debugging programs or helping other people to understand them, but of course, they take up valuable memory. (Then again, memory is getting cheaper all the time.)

MULTIPLE STATEMENTS ON ONE LINE

A program line may contain more than one statement, if the statements are separated by colons (":"). Using multiple statements on a single line improves the readability of the program by separating it into small blocks, and uses less memory for storing the program.

It is important to note that an IF statement will cause any statements appearing after it on the line to be ignored if the IF condition turns out to be false. This is the feature that allows a group of statements to be executed conditionally.

A multiple-statement line may be entered without a line number, but NIBL will only execute the first statement on the line, ignoring the rest.

POWERING UP

NIBL is capable of executing a program in ROM in Page 2 immediately upon powering up, without the need for the user to give the RUN command at the teletype. When NIBL initializes, it examines Page 2 and makes an educated guess about the possible existence of a legal NIBL program in that page. If NIBL thinks there really is a program there, it starts executing it immediately; thus, if the program halts for some reason, the value of PAGE will be 2. But if NIBL fails to find a legal program in Page 2 initially, it sets the value of PAGE to 1 (the normal case) and prompts at the teletype.

When executing programs, NIBL periodically checks for keyboard interrupt, returning to edit mode if it detects it. Therefore, if a NIBL program is to be executed with the teletype disconnected, the Sense B line of the SC/MP should be set high so that NIBL will not sense an interrupt while running. This would allow a NIBL system to act as a process controller which starts executing immediately upon powering up.

SEE MICROCOMPUTER SOFTWARE DEPOSITORY PROGRAM INDEX FOR TAPE COPIES OF THIS PROGRAM

```
     0001             LIST    1
               ;*****************************************************
               ;*                                                   *
               ;*     WE ARE TIED DOWN TO A LANGUAGE WHICH          *
               ;*     MAKES UP IN OBSCURITY WHAT IT LACKS           *
               ;*     IN STYLE.                                     *
               ;*                    -- TOM STOPPARD                *
               ;*                                                   *
               ;*****************************************************
     0020      TSTBIT  =       020             ;I. L.  INSTRUCTION FLAGS
     0040      JMPBIT  =       040
     0080      CALBIT  =       080
     0001      P1      =       1               ;SC/MP POINTER ASSIGNMENTS
     0002      P2      =       2
     0003      P3      =       3
     FF80      EREG    =       -128            ;THE EXTENSION REGISTER
               ; DISPLACEMENTS FOR RAM VARIABLES USED BY INTERPRETER
     FFFF      DOPTR   =       -1              ;DO-STACK POINTER
     FFFE      FORPTR  =       -2              ;FOR-STACK POINTER
     FFFD      LSTK    =       -3              ;ARITHMETIC STACK POINTER
     FFFC      SBRPTR  =       -4              ;GOSUB STACK POINTER
     FFFB      PCLOW   =       -5              ;I. L.  PROGRAM COUNTER
     FFFA      PCHIGH  =       -6
     FFF9      PCSTK   =       -7              ;I. L.  CALL STACK POINTER
     FFF8      LOLINE  =       -8              ;CURRENT LINE NUMBER
     FFF7      HILINE  =       -9
     FFF6      PAGE    =       -10             ;VALUE OF CURRENT PAGE
     FFF5      LISTNG  =       -11             ;LISTING FLAG
     FFF4      RUNMOD  =       -12             ;RUN/EDIT FLAG
     FFF3      LABLLO  =       -13
     FFF2      LABLHI  =       -14
     FFF1      P1LOW   =       -15             ;SPACE TO SAVE CURSOR
     FFF0      P1HIGH  =       -16
     FFEF      LO      =       -17
     FFEE      HI      =       -18
     FFED      FAILLO  =       -19
     FFEC      FAILHI  =       -20
     FFEB      NUM     =       -21
     FFEA      TEMP    =       -22
     FFE9      TEMP2   =       -23
     FFE8      TEMP3   =       -24
     FFE7      CHRNUM  =       -25
     FFE6      RNDF    =       -26
     FFE5      RNDX    =       -27             ;SEEDS FOR RANDOM NUMBER
     FFE4      RNDY    =       -28
               ; ALLOCATION OF RAM FOR NIBL VARIABLES, STACKS,
               ;  AND LINE BUFFER
 0000                  .=01000+28
 101C          VARS:   .=.+52                  ; NIBL VARIABLES A-Z
 1050          AESTK:  .=.+26                  ; ARITHMETIC STACK
 106A          SBRSTK: .=.+16                  ;GOSUB STACK
 107A          DOSTAK: .=.+16                  ;DO/UNTIL STACK
 108A          FORSTK: .=.+28                  ;FOR/NEXT STACK
 10A6          PCSTAK: .=.+48                  ;I. L.  CALL STACK
 10D6          LBUF:   .=.+74                  ;LINE BUFFER
 1120          PGM:    .=0                     ;USER'S PROGRAM
                       .MACRO  LDPI,P,VAL
                       .MLOC   TEMP
                       .SET    TEMP,VAL
                       LDI     H(TEMP)
                       XPAH    P
                       LDI     L(TEMP)
                       XPAL    P
                        ENDM
               ;*************************************
               ;*      INITIALIZATION OF NIBL       *
               ;*************************************
 0000 08               NOP
 0001                  LDPI    P2,VARS         ;POINT P2 AT VARIABLES
 0007                  LDPI    P1,PGM          ;POINT P1 AT PAGE ONE PROGRAM
 000D C4FF             LDI     -1              ;STORE -1 AT START OF PROGRAM
 000F C900             ST      0(P1)
 0011 C901             ST      1(P1)
 0013 C40D             LDI     0D              ;ALSO STORE A DUMMY END-OF-LI
 0015 C9FF             ST      -1(P1)
 0017 C402             LDI     2               ;POINT P2 AT PAGE 2,
 0019 CAF6             ST      PAGE(P2)        ; INITIALLY SET PAGE TO 2
 001B 31               XPAL    P1
 001C C420             LDI     020
 001E 35               XPAH    P1
 001F B902             DLD     2(P1)           ;CHECK IF THERE IS REALLY
 0021 01               XAE                     ; A PROGRAM IN PAGE 2:
 0022 C180             LD      EREG(P1)        ; IF FIRST LINE LENGTH
 0024 E40D             XRI     0D              ; POINTS TO CARR. RETURN
 0026 9802             JZ      $0              ; AT END OF LINE
 0028 BAF6             DLD     PAGE(P2)        ;IF NOT, PAGE = 1
 002A C420     $0:     LDI     020
 002C 35       $LOOP:  XPAH    P1
 002D C4FF             LDI     -1              ;STORE -1 IN 2 CONSECUTIVE
 002F C900             ST      (P1)            ; LOCATIONS AT START OF PAGE
 0031 C901             ST      1(P1)
 0033 C40D             LDI     0D              ;ALSO PUT A DUMMY END-OF-LINE
 0035 C9FF             ST      -1(P1)          ; JUST BEFORE TEXT
 0037 35               XPAH    P1              ;UPDATE P1 TO POINT TO
 0038 02               CCL                     ; NEXT PAGE (UNTIL PAGE=8)
 0039 F410             ADI     010             ;REPEAT INITIALIZATION
 003B E480             XRI     080             ; FOR PAGES 2-7
 003D 9804             JZ      $1
 003F E480             XRI     080
 0041 90E9             JMP     $LOOP
 0043 C400     $1:     LDI     0               ;CLEAR SOME FLAGS
 0045 CAF4             ST      RUNMOD(P2)
 0047 CAF5             ST      LISTNG(P2)
 0049 C454             LDI     L(BEGIN)        ;INITIALIZE IL PC SO THAT
 004B CAFB             ST      PCLOW(P2)       ; NIBL PROGRAM
 004D C40C             LDI     H(BEGIN)        ; IS EXECUTED IMMEDIATELY
 004F CAFA             ST      PCHIGH(P2)
 0051 C400     CLEAR:  LDI     0
 0053 CAEA             ST      TEMP(P2)
 0055 01               XAE
 0056 C400     CLEAR1: LDI     0               ;SET ALL VARIABLES
 0058 CA80             ST      EREG(P2)        ; TO ZERO
 005A AAEA             ILD     TEMP(P2)
 005C 01               XAE
 005D C434             LDI     52
 005F 60               XRE
 0060 9CF4             JNZ     CLEAR1
 0062 C450             LDI     L(AESTK)        ;INITIALIZE SOME STACKS:
 0064 CAFD             ST      LSTK(P2)        ; ARITHMETIC STACK,
 0066 C47A             LDI     L(DOSTAK)
 0068 CAFF             ST      DOPTR(P2)       ; DO/UNTIL STACK,
 006A C46A             LDI     L(SBRSTK)
 006C CAFC             ST      SBRPTR(P2)      ; GOSUB STACK,
 006E C4A6             LDI     L(PCSTAK)
 0070 CAF9             ST      PCSTK(P2)       ; I. L.  CALL STACK,
 0072 C48A             LDI     L(FORSTK)
 0074 CAFE             ST      FORPTR(P2)      ; FOR/NEXT STACK
               ;*************************************
               ;*   INTERMEDIATE LANGUAGE EXECUTOR  *
               ;*************************************
 0076 C2FB     EXECIL: LD      PCLOW(P2)       ;SET P3 TO CURRENT
 0078 33               XPAL    P3              ; IL PC.
 0079 C2FA             LD      PCHIGH(P2)
 007B 37               XPAH    P3
 007C C701     CHEAT:  LD      @1(P3)
 007E 01               XAE                     ;GET NEW IL INSTRUCTION
 007F C701             LD      @1(P3)          ; INTO P3 THROUGH
 0081 33               XPAL    P3              ; OBSCURE METHODS
 0082 CAFB             ST      PCLOW(P2)       ;SIMULTANEOUSLY, INCREMENT
 0084 40               LDE                     ; THE IL PC BY 2
 0085 37               XPAH    P3
 0086 CAFA             ST      PCHIGH(P2)
 0088 40               LDE
 0089 D4F0             ANI     0F0             ;CHECK IF IL INSTRUCTION
 008B E420             XRI     TSTBIT          ; IS A 'TEST'
 008D 9836             JZ      TST
 008F E4A0             XRI     CALBIT!TSTBIT   ;CHECK FOR IL CALL
 0091 980D             JZ      ILCALL
 0093 E4C0             XRI     JMPBIT!CALBIT   ;CHECK FOR IL JUMP
 0095 9C06             JNZ     NOJUMP
 0097 37               XPAH    P3              ;*** I. L.  JUMP ***
 0098 D40F             ANI     0F              ;ALL IT TAKES IS SCRUBBING
 009A 37               XPAH    P3              ; THE JUMP FLAG OFF OF P3
 009B 90DF     CHEAT1: JMP     CHEAT
 009D 3F       NOJUMP: XPPC    P3              ;MUST BE AN ML SUBROUTINE
 009E 90D6             JMP     EXECIL          ; IF NONE OF THE ABOVE
               ;*************************************
               ;*     INTERMEDIATE LANGUAGE CALL    *
               ;*************************************
 00A0 C2F9     ILCALL: LD      PCSTK(P2)
 00A2 E4D6             XRI     L(LBUF)         ;CHECK FOR STACK OVERFLOW
 00A4 9C04             JNZ     ILC1
 00A6 C40A             LDI     10
 00A8 9063             JMP     E0A
 00AA E4D6     ILC1:   XRI     L(LBUF)         ;RESTORE ACCUMULATOR
 00AC 33               XPAL    P3              ;SAVE LOW BYTE OF NEW
 00AD CAEA             ST      TEMP(P2)        ; I. L.  PC IN TEMP
 00AF C410             LDI     H(PCSTAK)       ;POINT P3 AT I. L.
 00B1 37               XPAH    P3              ; SUBROUTINE STACK
 00B2 C2FB             LD      PCLOW(P2)       ;SAVE OLD I. L.  PC ON STACK
 00B4 CF01             ST      @1(P3)
 00B6 C2FA             LD      PCHIGH(P2)
 00B8 CF01             ST      @1(P3)
 00BA C2EA             LD      TEMP(P2)        ;GET LOW BYTE OF NEW
 00BC 33               XPAL    P3              ; I. L.  PC INTO P3 LOW
 00BD CAF9             ST      PCSTK(P2)       ;UPDATE I. L.  STACK POINTER
 00BF 40               LDE                     ;GET HIGH BYTE OF NEW P3
 00C0 D40F             ANI     0F              ; I. L.  PC INTO P3 HIGH
 00C2 37               XPAH    P3
 00C3 90B7             JMP     CHEAT
               ;*************************************
               ;*      I. L.  'TEST' INSTRUCTION    *
               ;*************************************
                        LOCAL
 00C5 CAE7     TST:    ST      CHRNUM(P2)      ;CLEAR NUMBER OF CHARS SCANNE
 00C7 C501     $SCAN:  LD      @1(P1)          ;SLEW OFF SPACES
 00C9 E420             XRI     ' '
 00CB 98FA             JZ      $SCAN
 00CD C5FF             LD      @-1(P1)         ;REPOSITION CURSOR
 00CF C2FA             LD      PCHIGH(P2)      ;POINT P3 AT IL TABLE
 00D1 37               XPAH    P3
 00D2 D40F             ANI     0F              ;FAIL ADDRESS <- OLD P3
 00D4 CAEC             ST      FAILHI(P2)
 00D6 C2FB             LD      PCLOW(P2)
 00D8 33               XPAL    P3
 00D9 CAED             ST      FAILLO(P2)
 00DB C701     $LOOP:  LD      @1(P3)
 00DD 01               XAE                     ;SAVE CHAR FROM TABLE
 00DE BAE7             DLD     CHRNUM(P2)      ;DECREMENT CHAR COUNT
 00E0 40               LDE                     ;GET CHAR BACK
 00E1 D47F             ANI     07F             ;SCRUB OFF FLAG (IF ANY)
 00E3 E501             XOR     @1(P1)          ;IS CHAR EQUAL TO TEXT CHAR?
 00E5 9C07             JNZ     $NEQ            ;NO - END TEST
 00E7 40               LDE                     ;YES - BUT IS IT LAST CHAR?
 00E8 94F1             JP      $LOOP           ;IF NOT, CONTINUE TO COMPARE
 00EA 9090             JMP     CHEAT           ;IF SO, GET NEXT I. L.
 00EC 9088     X0:     JMP     EXECIL          ; INSTRUCTION
 00EE C2E7     $NEQ:   LD      CHRNUM(P2)      ;RESTORE P1 TO
 00F0 01               XAE                     ; ORIGINAL VALUE
 00F1 C580             LD      @EREG(P1)
 00F3 C2ED             LD      FAILLO(P2)      ;LOAD TEST-FAIL ADDRESS
 00F5 33               XPAL    P3              ; INTO P3
 00F6 C2EC             LD      FAILHI(P2)
 00F8 37               XPAH    P3
 00F9 90A0             JMP     CHEAT1          ;GET NEXT IL INSTRUCTION
               ;*************************************
               ;*      I. L.  SUBROUTINE RETURN     *
               ;*************************************
 00FB C410     RTN:    LDI     H(PCSTAK)       ;POINT P3 AT I. L.  PC STACK
 00FD 37               XPAH    P3
 00FE C2F9             LD      PCSTK(P2)
 0100 33               XPAL    P3
```

```
247 0101 C7FF           LD      @-1(P3)         ;GET HIGH PART OF OLD PC
248 0103 01             XAE
249 0104 C7FF           LD      @-1(P3)         ;GET LOW PART OF OLD PC
250 0106 33             XPAL    P3
251 0107 CAF9           ST      PCSTK(P2)       ;UPDATE IL STACK POINTER
252 0109 40             LDE
253 010A 37             XPAH    P3              ;P3 NOW HAS OLD IL PC
254 010B 908E           JMP     CHEAT1
255 010D 9041   EOA:    JMP     E0
256
257
258             ;*************************************
259             ;*      SAVE GOSUB RETURN ADDRESS      *
260             ;*************************************
261
262 010F C2FC   SAV:    LD      SBRPTR(P2)
263 0111 E47A           XRI     L(DOSTAK)       ;CHECK FOR MORE
264 0113 981C           JZ      SAV2            ; THAN 8 SAVES
265 0115 AAFC           ILD     SBRPTR(P2)
266 0117 AAFC           ILD     SBRPTR(P2)
267 0119 33             XPAL    P3              ;SET P3 TO
268 011A C410           LDI     H(SBRSTK)       ; SUBROUTINE STACK TOP.
269 011C 37             XPAH    P3
270 011D C2F4           LD      RUNMOD(P2)      ;IF IMMEDIATE MODE,
271 011F 980A           JZ      SAV1            ; SAVE NEGATIVE ADDRESS.
272 0121 35             XPAH    P1              ;SAVE HIGH PORTION
273 0122 CBFF           ST      -1(P3)          ; OF CURSOR
274 0124 35             XPAH    P1
275 0125 31             XPAL    P1              ;SAVE LOW PORTION
276 0126 CBFE           ST      -2(P3)          ; OF CURSOR
277 0128 31             XPAL    P1
278 0129 90C1           JMP     X0              ;RETURN
279 012B C4FF   SAV1:   LDI     -1              ;IMMEDIATE MODE
280 012D CBFF           ST      -1(P3)          ; RETURN ADDRESS IS
281 012F 90BB           JMP     X0              ; NEGATIVE.
282 0131 C40A   SAV2:   LDI     10              ;ERROR: MORE THAN
283 0133 901B           JMP     E0              ; 8 GOSUBS
284
285
286             ;*************************************
287             ;*     CHECK STATEMENT FINISHED      *
288             ;*************************************
289
290 0135 C501   DONE:   LD      @1(P1)          ;SKIP SPACES
291 0137 E420           XRI     ' '
292 0139 98FA           JZ      DONE
293 013B E42D           XRI     ' '+80H         ;IS IT CARRIAGE RETURN?
294 013D 9804           JZ      DONE1           ;YES - RETURN
295 013F E437           XRI     037             ;IS CHAR A ':' ?
296 0141 9C01           JNZ     DONE2           ;NO - ERROR
297 0143 3F     DONE1:  XPPC    P3              ;YES - RETURN
298 0144 C404   DONE2:  LDI     4
299 0146 9008           JMP     E0
300
301
302             ;*************************************
303             ;*     RETURN FROM GOSUB             *
304             ;*************************************
305
306 0148 C2FC   RSTR:   LD      SBRPTR(P2)
307 014A E46A           XRI     L(SBRSTK)       ;CHECK FOR RETURN
308 014C 9C04           JNZ     RSTR1           ; W/O GOSUB.
309 014E C409           LDI     9
310 0150 9043   E0:     JMP     E1              ;GOTO ERROR.
311 0152 BAFC   RSTR1:  DLD     SBRPTR(P2)
312 0154 BAFC           DLD     SBRPTR(P2)      ;POP GOSUB STACK,
313 0156 33             XPAL    P3              ; PUT PTR INTO P3.
314 0157 C410           LDI     H(SBRSTK)
315 0159 37             XPAH    P3
316 015A C301           LD      1(P3)           ;IF ADDRESS NEGATIVE,
317 015C 9409           JP      RSTR2           ; SUBROUTINE WAS CALLED
318 015E C402           JS      P3,FIN          ; IN IMMEDIATE MODE,
319 0165 9085   X1:     JMP     X0              ; SO FINISH UP EXECUTING
320 0167 35     RSTR2:  XPAH    P1              ;RESTORE CURSOR HIGH
321 0168 C300           LD      0(P3)
322 016A 31             XPAL    P1              ;RESTORE CURSOR LOW
323 016B C401           LDI     1               ;SET RUN MODE
324 016D CAF4           ST      RUNMOD(P2)
325 016F 90F4           JMP     X1
326
327
328             ;*************************************
329             ;*     TRANSFER TO NEW STATEMENT     *
330             ;*************************************
331
332 0171 C2F2   XFER:   LD      LABLHI(P2)      ;CHECK FOR NON-EXISTENT LINE
333 0173 9404           JP      XFER1
334 0175 C408           LDI     8
335 0177 901C           JMP     E1
336 0179 C401   XFER1:  LDI     1               ;SET RUN MODE TO 1
337 017B CAF4           ST      RUNMOD(P2)
338 017D 3F             XPPC    P3
339
340
341             ;*************************************
342             ;*     PRINT STRING IN TEXT          *
343             ;*************************************
344
345 017E        PRS:    LDPI    P3,PUTC-1       ;POINT P3 AT PUTC ROUTINE
346 0184 C501           LD      @1(P1)          ;LOAD NEXT CHAR
347 0186 E422           XRI     '"'             ;IF ", END OF
348 0188 98DB           JZ      X1              ; STRING
349 018A E42F           XRI     02F             ;IF CR, ERROR
350 018C 9805           JZ      PRS1
351 018E E40D           XRI     0D              ;RESTORE CHAR
352 0190 3F             XPPC    P3              ;PRINT CHAR
353 0191 90EB           JMP     PRS             ;GET NEXT CHAR
354 0193 C407   PRS1:   LDI     7               ;SYNTAX ERROR
355 0195 9035   E1:     JMP     E2
356
357
358             ;*************************************
359             ;*     PRINT NUMBER ON STACK         *
360             ;*************************************
361
362             ; THIS ROUTINE IS BASED ON DENNIS ALLISON'S BINARY TO DECIMAL
363             ; CONVERSION ROUTINE IN VOL. 1, #1 OF "DR. DOBB'S JOURNAL",
364             ; BUT IS MUCH MORE OBSCURE BECAUSE OF THE STACK MANIPULATION.
365
366                     .LOCAL
367 0197 C410   PRN:    LDI     H(AESTK)        ;POINT P3 AT A.E. STACK
368 0199 37             XPAH    P3
369 019A AAFD           ILD     LSTK(P2)
370 019C AAFD           ILD     LSTK(P2)
371 019E 33             XPAL    P3
372 019F C40A           LDI     10              ;PUT 10 ON STACK (WE'LL BE
373 01A1 CBFE           ST      -2(P3)          ; DIVIDING BY IT LATER)
374 01A3 C400           LDI     0
375 01A5 CBFF           ST      -1(P3)
376 01A7 C405           LDI     5               ;SET CHRNUM TO POINT TO PLACE
377 01A9 CAE7           ST      CHRNUM(P2)      ; IN STACK WHERE WE STORE
378 01AB C4FF           LDI     -1              ; THE CHARACTERS TO PRINT
379 01AD CB05           ST      5(P3)           ;FIRST CHAR IS A FLAG (-1)
380 01AF C3FD           LD      -3(P3)          ;CHECK IF NUMBER IS NEGATIVE
381 01B1 9413           JP      $1
382 01B3 C42D           LDI     '-'             ;PUT '-' ON STACK, AND NEGATE
383 01B5 CB04           ST      4(P3)           ; THE NUMBER
384 01B7 C400           LDI     0
385 01B9 03             SCL
386 01BA FBFC           CAD     -4(P3)
387 01BC CBFC           ST      -4(P3)
388 01BE C400           LDI     0
389 01C0 FBFD           CAD     -3(P3)
390 01C2 CBFD           ST      -3(P3)
391 01C4 909F           JMP     X1              ;GO DO DIVISION BY 10
392 01C6 C420   $1:     LDI     ' '             ;IF POSITIVE, PUT ' ' ON
393 01C8 CB04           ST      4(P3)           ; STACK BEFORE DIVISION
394 01CA 9099   X4:     JMP     X1
395 01CC 9057   E2:     JMP     ERR1
396
397             ; THE DIVISION IS PERFORMED, THEN CONTROL IS TRANSFERRED
398             ; TO PRN1, WHICH FOLLOWS.
399
400 01CE AAFD   PRN1:   ILD     LSTK(P2)        ;POINT P1 AT A.E. STACK
401 01D0 AAFD           ILD     LSTK(P2)
402 01D2 31             XPAL    P1
403 01D3 C410           LDI     H(AESTK)
404 01D5 35             XPAH    P1
405 01D6 AAE7           ILD     CHRNUM(P2)      ;INCREMENT CHARACTER STACK
406 01D8 01             XAE                     ; POINTER, PUT IN EX. REG.
407 01D9 C101           LD      1(P1)           ;GET REMAINDER FROM DIVIDE,
408 01DB DC30           ORI     '0'
409 01DD C980           ST      EREG(P1)        ; PUT IT ON THE STACK
410 01DF C1FD           LD      -3(P1)          ;IS THE QUOTIENT ZERO YET?
411 01E1 D9FC           OR      -4(P1)          ;
412 01E3 980A           JZ      $PRNT           ;YES - GO PRINT THE NUMBER
413 01E5 C40F           LDI     H(PRNUM1)       ;NO - CHANGE THE I.L. PC
414 01E7 CAFA           ST      PCHIGH(P2)      ; SO THAT DIVIDE IS
415 01E9 C42F           LDI     L(PRNUM1)       ; PERFORMED AGAIN
416 01EB CAFB           ST      PCLOW(P2)
417 01ED 90DB           JMP     X4              ;GO DO DIVISION BY 10 AGAIN
418 01EF        $PRNT:  LDPI    P3,PUTC-1       ;POINT P3 AT PUTC ROUTINE
419 01F5 C2F5           LD      LISTNG(P2)      ;IF LISTING, SKIP PRINTING
420 01F7 9C06           JNZ     $2              ; LEADING SPACE
421 01F9 C104           LD      4(P1)           ;PRINT EITHER '-'
422 01FB 3F             XPPC    P3              ; OR LEADING SPACE
423 01FC C2E7           LD      CHRNUM(P2)      ;GET EX. REG. VALUE BACK
424 01FE 01             XAE
425 01FF C580   $2:     LD      @EREG(P1)       ;POINT P3 AT FIRST CHAR
426 0201 C100           LD      (P1)            ; TO BE PRINTED
427 0203 3F     $LOOP:  XPPC    P3              ;PRINT THE CHARACTER
428 0204 C5FF           LD      @-1(P1)         ;GET NEXT CHARACTER
429 0206 94FB           JP      $LOOP           ;REPEAT UNTIL = -1
430 0208 C450           LDI     L(AESTK)
431 020A CAFD           ST      LSTK(P2)        ;CLEAR THE A.E. STACK
432 020C C2F5           LD      LISTNG(P2)      ;PRINT A TRAILING SPACE
433 020E 9CBA           JNZ     X4              ; IF NOT LISTING PROGRAM
434 0210 C420           LDI     ' '
435 0212 3F             XPPC    P3
436 0213 90B5           JMP     X4
437
438
439             ;*************************************
440             ;*     CARRIAGE RETURN/LINE FEED     *
441             ;*************************************
442
443 0215        NLINE:  LDPI    P3,PUTC-1       ;POINT P3 AT PUTC ROUTINE
444 021B C40D           LDI     0D              ;CARRIAGE RETURN
445 021D 3F             XPPC    P3
446 021E C40A           LDI     0A              ;LINE FEED
447 0220 3F             XPPC    P3
448 0221 90A7   X5:     JMP     X4
449
450
451             ;*************************************
452             ;*     ERROR ROUTINE                 *
453             ;*************************************
454
455                     .LOCAL
456 0223 C405   ERR:    LDI     5               ;SYNTAX ERROR
457 0225 CAEB   ERR1:   ST      NUM(P2)         ;SAVE ERROR #
458 0227 C2EB   ERR2:   LD      NUM(P2)
459 0229 CAEA           ST      TEMP(P2)
460 022B        .       LDPI    P3,PUTC-1       ;POINT P3 AT PUTC
461 0231 C40D           LDI     0D              ;PRINT CR/LF
462 0233 3F             XPPC    P3
463 0234 C40A           LDI     0A
464 0236 3F             XPPC    P3
465 0237        .       LDPI    P1,MESGS        ;P1 -> ERROR MESSAGES
466 023D BAEB   $1:     DLD     NUM(P2)         ;IS THIS THE RIGHT MESSAGE?
467 023F 9806           JZ      $MSG            ;YES - GO PRINT IT
468 0241 C501   $LOOP:  LD      @1(P1)          ;NO - SCAN THROUGH TO
469 0243 94FC           JP      $LOOP           ; NEXT MESSAGE
470 0245 90F6           JMP     $1
471 0247 C501   $MSG:   LD      @1(P1)          ;GET MESSAGE CHAR
472 0249 3F             XPPC    P3              ;PRINT IT
473 024A C1FF           LD      -1(P1)          ;IS MESSAGE DONE?
474 024C 94F9           JP      $MSG            ;NO - GET NEXT CHAR
475 024E C2EA           LD      TEMP(P2)        ;WAS THIS A BREAK MESSAGE?
476 0250 E40E           XRI     14
477 0252 980D           JZ      $3              ;YES - SKIP PRINTING 'ERROR'
478 0254        .       LDPI    P1,MESGS        ;NO - PRINT 'ERROR'
479 025A C501   $2:     LD      @1(P1)          ;GET CHARACTER
480 025C 3F             XPPC    P3              ;PRINT IT
481 025D C1FF           LD      -1(P1)          ;DONE?
482 025F 94F9           JP      $2              ;NO - REPEAT LOOP
483 0261 C2F4   $3:     LD      RUNMOD(P2)      ;DON'T PRINT LINE #
484 0263 984D           JZ      FIN             ; IF IMMEDIATE MODE
485 0265 C420           LDI     ' '
486 0267 3F             XPPC    P3              ;SPACE
487 0268 C441           LDI     'A'             ;AT
488 026A 3F             XPPC    P3
489 026B C454           LDI     'T'
490 026D 3F             XPPC    P3
491 026E C410           LDI     H(AESTK)        ;POINT P3 AT A.E. STACK
492 0270 37             XPAH    P3
493 0271 AAFD           ILD     LSTK(P2)
494 0273 AAFD           ILD     LSTK(P2)
495 0275 33             XPAL    P3
496 0276 C2F7           LD      HILINE(P2)      ;GET HIGH BYTE OF LINE #
497 0278 CBFF           ST      -1(P3)          ;PUT ON STACK
498 027A C2F8           LD      LOLINE(P2)      ;GET LOW BYTE OF LINE #
499 027C CBFE           ST      -2(P3)          ;PUT ON STACK
500 027E C42D           LDI     L(ERRNUM)       ;GO TO PRN
501 0280 CAFB           ST      PCLOW(P2)
502 0282 C40E           LDI     H(ERRNUM)
503 0284 CAFA           ST      PCHIGH(P2)
504 0286 9099   X5A:    JMP     X5
505
506
507             ;*************************************
508             ;*     BREAK, NXT, FIN, & STRT       *
509             ;*************************************
510
```

```
0288 C40E   BREAK:  LDI     14
028A 9099   E3A:    JMP     ERR1
                                        ;*** NEXT STATEMENT ***
028C C2F4   NXT:    LD      RUNMOD(P2)  ;IF IN IMMED. MODE,
028E 9822           JZ      FIN         ; STOP EXECUTION
0290 C100           LD      (P1)        ;IF WE HIT END OF FILE,
0292 D480           ANI     080         ; FINISH UP THINGS
0294 9C1C           JNZ     FIN
0296 06             CSA                 ;BREAK IF SOMEONE IS
0297 D420           ANI     020         ; TYPING ON THE CONSOLE
0299 98ED           JZ      BREAK
029B C1FF           LD      -1(P1)      ;GET LAST CHARACTER SCANNED
029D E40D           XRI     0D          ;WAS IT CARRIAGE RETURN?
029F 9C08           JNZ     NXT1        ;YES - SKIP FOLLOWING UPDATES
02A1 C501           LD      @1(P1)      ;GET HIGH BYTE OF NEXT LINE #
02A3 CAF7           ST      HILINE(P2)  ;SAVE IT
02A5 C502           LD      @2(P1)      ;GET LOW BYTE OF LINE #, SKIP
02A7 CAF8           ST      LOLINE(P2)  ; LINE LENGTH BYTE
02A9 C40C   NXT1:   LDI     H(STMT)     ;GO TO 'STMT' IN IL TABLE
02AB CAFA           ST      PCHIGH(P2)
02AD C482           LDI     L(STMT)
02AF CAFB           ST      PCLOW(P2)
02B1 3F             XPPC    P3

02B2 C400   FIN:    LDI     0           ;*** FINISH EXECUTION ***
02B4 CAF4           ST      RUNMOD(P2)  ;CLEAR RUN MODE
02B6 C450           LDI     L(AESTK)    ;CLEAR ARITHMETIC STACK
02B8 CAFD           ST      LSTK(P2)
02BA C418           LDI     L(START)    ;SET IL PC TO GETTING LINES
02BC CAFB           ST      PCLOW(P2)

02BE C40C           LDI     H(START)
02C0 CAFA           ST      PCHIGH(P2)
02C2 C4A6           LDI     L(PCSTAK)
02C4 CAF9           ST      PCSTK(P2)
02C6 90BE           JMP     X5A
                                        ;*** START EXECUTION ***
02C8 AAF4   STRT:   ILD     RUNMOD(P2)  ;RUN MODE = 1
02CA C2E9           LD      TEMP2(P2)   ;POINT CURSOR TO
02CC 35             XPAH    P1          ; START OF NIBL PROGRAM
02CD C2E8           LD      TEMP3(P2)
02CF 31             XPAL    P1
02D0 C46A           LDI     L(SBRSTK)   ;EMPTY SOME STACKS:
02D2 CAFC           ST      SBRPTR(P2)  ; GOSUB STACK,
02D4 C48A           LDI     L(FORSTK)
02D6 CAFE           ST      FORPTR(P2)  ; FOR STACK
02D8 C47A           LDI     L(DOSTAK)
02DA CAFF           ST      DOPTR(P2)   ; & DO/UNTIL STACK
02DC 3F             XPPC    P3          ;RETURN
02DD 90A7   X6:     JMP     X5A
02DF 90A9   E4:     JMP     E3A


                    ;*************************************
                    ;*       LIST NIBL PROGRAM           *
                    ;*************************************

02E1 C100   LST:    LD      (P1)        ;CHECK FOR END OF FILE
02E3 E480           XRI     080
02E5 9418           JP      LST2
02E7 C410           LDI     H(AESTK)    ;GET LINE NUMBER ONTO STACK
02E9 37             XPAH    P3
02EA AAFD           ILD     LSTK(P2)
02EC AAFD           ILD     LSTK(P2)
02EE 33             XPAL    P3
02EF C501           LD      @1(P1)
02F1 CBFF           ST      -1(P3)
02F3 C501           LD      @1(P1)
02F5 CBFE           ST      -2(P3)
02F7 C501           LD      @1(P1)      ;SKIP OVER LINE LENGTH
02F9 C401           LDI     1
02FB CAF5           ST      LISTNG(P2)  ;SET LISTING FLAG
02FD 90DE           JMP     X6          ;GO PRINT LINE NUMBER
02FF C400   LST2:   LDI     0
0301 CAF5           ST      LISTNG(P2)  ;CLEAR LISTING FLAG
0303 C402           JS      P3,NXT      ;GO TO NXT
030A 90D1   X6A:    JMP     X6
030C 90D1   E5:     JMP     E4
030E        LST3:   LDPI    P3,PUTC-1   ;POINT P3 AT PUTC
0314 06     LST4:   CSA
0315 D420           ANI     020
0317 98E6           JZ      LST2        ;IF TYPING, STOP
0319 C501           LD      @1(P1)      ;GET NEXT CHAR
031B E40D           XRI     0D          ;TEST FOR CR
031D 9805           JZ      LST5

031F E40D           XRI     0D          ;GET CHARACTER
0321 3F             XPPC    P3          ;PRINT CHARACTER
0322 90F0           JMP     LST4
0324 C40D   LST5:   LDI     0D          ;CARRIAGE RETURN
0326 3F             XPPC    P3
0327 C40A           LDI     0A          ;LINE FEED
0329 3F             XPPC    P3
032A 02             CCL
032B C447           LDI     L(LIST3)
032D CAFB           ST      PCLOW(P2)
032F C40C           LDI     H(LIST3)
0331 CAFA           ST      PCHIGH(P2)
0333 90AC           JMP     LST         ;GET NEXT LINE


                    ;*************************************
                    ;*          ADD AND SUBTRACT         *
                    ;*************************************

0335 C410   ADD:    LDI     H(AESTK)    ;SET P3 TO CURRENT
0337 37             XPAH    P3          ; STACK LOCATION
0338 BAFD           DLD     LSTK(P2)
033A BAFD           DLD     LSTK(P2)
033C 33             XPAL    P3
033D 02             CCL
033E C3FE           LD      -2(P3)      ;REPLACE TWO TOP ITEMS
0340 F300           ADD     0(P3)       ; ON STACK BY THEIR SUM
0342 CBFE           ST      -2(P3)
0344 C3FF           LD      -1(P3)
0346 F301           ADD     1(P3)
0348 CBFF           ST      -1(P3)
034A 90BE   X7:     JMP     X6A

034C C410   SUB:    LDI     H(AESTK)    ;SET P3 TO CURRENT
034E 37             XPAH    P3          ; STACK LOCATION
034F BAFD           DLD     LSTK(P2)
0351 BAFD           DLD     LSTK(P2)
0353 33             XPAL    P3
0354 03             SCL
0355 C3FE           LD      -2(P3)      ;REPLACE TWO TOP ITEMS
0357 FB00           CAD     0(P3)       ; ON STACK BY THEIR DIFFERENC
0359 CBFE           ST      -2(P3)
035B C3FF           LD      -1(P3)
035D FB01           CAD     1(P3)
035F CBFF           ST      -1(P3)
0361 90A7           JMP     X6A


                    ************************************
```

```
                    ;*************************************
                    ;*          NEGATE                   *
                    ;*************************************

0363 C410   NEG:    LDI     H(AESTK)    ;SET P3 TO CURRENT
0365 37             XPAH    P3          ; STACK LOCATION

0366 C2FD           LD      LSTK(P2)
0368 33             XPAL    P3
0369 03             SCL
036A C400           LDI     0
036C FBFE           CAD     -2(P3)      ;NEGATE TOP ITEM ON STACK
036E CBFE           ST      -2(P3)
0370 C400           LDI     0
0372 FBFF           CAD     -1(P3)
0374 CBFF           ST      -1(P3)
0376 90D2   X8:     JMP     X7
0378 9092   E6:     JMP     E5


                    ;*************************************
                    ;*          MULTIPLY                 *
                    ;*************************************

                    LOCAL
037A C410   MUL:    LDI     H(AESTK)    ;SET P3 TO CURRENT
037C 37             XPAH    P3          ; STACK LOCATION
037D C2FD           LD      LSTK(P2)
037F 33             XPAL    P3          ;DETERMINE SIGN OF PRODUCT,
0380 C3FF           LD      -1(P3)      ; SAVE IN TEMP(P2)
0382 E3FD           XOR     -3(P3)
0384 CAEA           ST      TEMP(P2)
0386 C3FF           LD      -1(P3)      ;CHECK FOR NEGATIVE
0388 940D           JP      $1          ; MULTIPLIER
038A 03             SCL
038B C400           LDI     0           ;IF NEGATIVE,
038D FBFE           CAD     -2(P3)      ; NEGATE
038F CBFE           ST      -2(P3)
0391 C400           LDI     0
0393 FBFF           CAD     -1(P3)
0395 CBFF           ST      -1(P3)
0397 C3FD   $1:     LD      -3(P3)      ;CHECK FOR NEGATIVE
0399 940D           JP      $2          ; MULTIPLICAND
039B 03             SCL
039C C400           LDI     0           ;IF NEGATIVE,
039E FBFC           CAD     -4(P3)      ; NEGATE
03A0 CBFC           ST      -4(P3)
03A2 C400           LDI     0
03A4 FBFD           CAD     -3(P3)
03A6 CBFD           ST      -3(P3)
03A8 C400   $2:     LDI     0           ;CLEAR WORKSPACE
03AA CB00           ST      0(P3)
03AC CB01           ST      1(P3)
03AE CB02           ST      2(P3)
03B0 CB03           ST      3(P3)
03B2 C410           LDI     16          ;SET COUNTER TO 16
03B4 CAEB           ST      NUM(P2)
03B6 C3FF   $LOOP:  LD      -1(P3)      ;ROTATE MULTIPLIER
03B8 1F             RRL                 ; RIGHT ONE BIT
03B9 CBFF           ST      -1(P3)
03BB C3FE           LD      -2(P3)
03BD 1F             RRL
03BE CBFE           ST      -2(P3)
03C0 06             CSA                 ;CHECK FOR CARRY BIT
03C1 9411           JP      $3          ;IF NOT SET, DON'T DO ADD
03C3 02             CCL
03C4 C302           LD      2(P3)       ;ADD MULTIPLICAND
03C6 F3FC           ADD     -4(P3)      ; INTO WORKSPACE
03C8 CB02           ST      2(P3)
03CA C303           LD      3(P3)
03CC F3FD           ADD     -3(P3)
03CE CB03           ST      3(P3)
03D0 9002           JMP     $3
03D2 90A4   E6A:    JMP     E6
03D4 02     $3:     CCL
03D5 C303           LD      3(P3)       ;SHIFT WORKSPACE RIGHT BY 1
03D7 1F             RRL
03D8 CB03           ST      3(P3)
03DA C302           LD      2(P3)
03DC 1F             RRL
03DD CB02           ST      2(P3)
03DF C301           LD      1(P3)
03E1 1F             RRL
03E2 CB01           ST      1(P3)
03E4 C300           LD      0(P3)
03E6 1F             RRL
03E7 CB00           ST      0(P3)
03E9 BAEB           DLD     NUM(P2)     ;DECREMENT COUNTER
03EB 9CC9           JNZ     $LOOP       ;LOOP IF NOT ZERO
03ED 9002           JMP     $4
03EF 9085   X9:     JMP     X8
03F1 C2EA   $4:     LD      TEMP(P2)    ;CHECK SIGN WORD
03F3 940D           JP      $EXIT       ;IF BIT7 = 1, NEGATE PRODUCT
03F5 03             SCL
03F6 C400           LDI     0
03F8 FB00           CAD     0(P3)
03FA CB00           ST      0(P3)
03FC C400           LDI     0
03FE FB01           CAD     1(P3)
0400 CB01           ST      1(P3)
0402 C300   $EXIT:  LD      0(P3)       ;PUT PRODUCT ON TOP
0404 CBFC           ST      -4(P3)      ; OF STACK
0406 C301           LD      1(P3)
0408 CBFD           ST      -3(P3)
040A BAFD           DLD     LSTK(P2)    ;SUBTRACT 2 FROM
040C BAFD           DLD     LSTK(P2)    ; LSTK
040E 90DF           JMP     X9


                    ;*************************************
                    ;*          DIVIDE                   *
                    ;*************************************

                    LOCAL
0410 C410   DIV     LDI     H(AESTK)
0412 37             XPAH    P3
0413 C2FD           LD      LSTK(P2)
0415 33             XPAL    P3
0416 C3FF           LD      -1(P3)      ;CHECK FOR DIVISION BY 0
0418 DBFE           OR      -2(P3)
041A 9C04           JNZ     $0
041C C40D           LDI     13
041E 90B2           JMP     E6A
0420 C3FD   $0:     LD      -3(P3)
0422 E3FF           XOR     -1(P3)
0424 CAEA           ST      TEMP(P2)    ;SAVE SIGN OF QUOTIENT
0426 C3FD           LD      -3(P3)      ;IS DIVIDEND POSITIVE?
0428 9411           JP      $POS        ;YES - JUMP
042A C400           LDI     0
042C 03             SCL
042D FBFC           CAD     -4(P3)      ;NO - NEGATE DIVIDEND,
042F CB03           ST      3(P3)       ; STORE IN RIGHT HALF
0431 C400           LDI     0           ; OF 32-BIT ACCUMULATOR
0433 FBFD           CAD     -3(P3)
0435 CB02           ST      2(P3)
0437 900A           JMP     $1
```

```
0439 90B4   X9A:    JMP     X9
043B C3FD   $POS:   LD      -3(P3)          ;STORE NON-NEGATED DIVIDEND
043D CB02           ST      2(P3)           ;  IN 32-BIT ACCUMULATOR
043F C3FC           LD      -4(P3)
0441 CB03           ST      3(P3)
0443 C3FF   $1:     LD      -1(P3)          ;CHECK FOR NEGATIVE DIVISOR
0445 940D           JP      $2
0447 C400           LDI     0               ;NEGATE DIVISOR
0449 03             SCL
044A FBFE           CAD     -2(P3)
044C CBFE           ST      -2(P3)
044E C400           LDI     0
0450 FBFF           CAD     -1(P3)
0452 CBFF           ST      -1(P3)
0454 C400   $2:     LDI     0               ;PUT ZERO IN:
0456 CB01           ST      1(P3)           ;  LEFT HALF OF 32-BIT ACC,
0458 CB00           ST      0(P3)
045A CAEB           ST      NUM(P2)         ;  THE COUNTER, AND
045C CBFD           ST      -3(P3)          ;  IN THE DIVIDEND, NOW USED
045E CBFC           ST      -4(P3)          ;  STORE THE QUOTIENT
0460 02     $LOOP:  CCL                     ;BEGIN MAIN DIVIDE LOOP:
0461 C3FC           LD      -4(P3)          ;  SHIFT QUOTIENT LEFT,
0463 F3FC           ADD     -4(P3)
0465 CBFC           ST      -4(P3)
0467 C3FD           LD      -3(P3)
0469 F3FD           ADD     -3(P3)
046B CBFD           ST      -3(P3)
046D 02             CCL                     ;  SHIFT 32-BIT ACC LEFT,
046E C303           LD      3(P3)
0470 F303           ADD     3(P3)
0472 CB03           ST      3(P3)
0474 C302           LD      2(P3)
0476 F302           ADD     2(P3)
0478 CB02           ST      2(P3)
047A C301           LD      1(P3)
047C F301           ADD     1(P3)
047E CB01           ST      1(P3)
0480 C300           LD      (P3)
0482 F300           ADD     (P3)
0484 CB00           ST      (P3)
0486 03             SCL
0487 C301           LD      1(P3)           ;  SUBTRACT DIVISOR INTO
0489 FBFE           CAD     -2(P3)          ;   LEFT HALF OF ACC,
048B CB01           ST      1(P3)
048D C300           LD      (P3)
048F FBFF           CAD     -1(P3)
0491 CB00           ST      (P3)
0493 9411           JP      $ENT1           ;  IF RESULT IS NEGATIVE,
0495 02             CCL                     ;   RESTORE ORIGINAL CONTENTS
0496 C301           LD      1(P3)           ;   OF ACC BY ADDING DIVISOR
0498 F3FE           ADD     -2(P3)
049A CB01           ST      1(P3)
049C C300           LD      (P3)
049E F3FF           ADD     -1(P3)
04A0 CB00           ST      (P3)
04A2 9008           JMP     $3
04A4 9093   X9B:    JMP     X9A
04A6 C3FC   $ENT1:  LD      -4(P3)          ;  ELSE IF RESULT POSITIVE,
04A8 DC01           ORI     1               ;   RECORD A 1 IN QUOTIENT
04AA CBFC           ST      -4(P3)          ;   W/O RESTORING THE ACC
04AC AAEB   $3:     ILD     NUM(P2)         ;  INCREMENT THE COUNTER
04AE E410           XRI     16              ;  ARE WE DONE?
04B0 9CAE           JNZ     $LOOP           ;   LOOP IF NOT DONE
04B2 C2EA           LD      TEMP(P2)        ;CHECK THE QUOTIENT'S SIGN,
04B4 940D           JP      $END            ;  NEGATING IF, NECESSARY
04B6 C400           LDI     0
04B8 03             SCL
04B9 FBFC           CAD     -4(P3)
04BB CBFC           ST      -4(P3)
04BD C400           LDI     0
04BF FBFD           CAD     -3(P3)
04C1 CBFD           ST      -3(P3)
04C3 BAFD   $END:   DLD     LSTK(P2)        ;DECREMENT THE STACK POINTER,
04C5 BAFD           DLD     LSTK(P2)
04C7 90DB           JMP     X9B             ;  AND EXIT


                    ;*************************************
                    ;*          STORE VARIABLE           *
                    ;*************************************

04C9 C410   STORE:  LDI     H(AESTK)        ;SET P3 TO STACK
04CB 37             XPAH    P3
04CC C2FD           LD      LSTK(P2)
04CE 33             XPAL    P3
04CF C7FD           LD      @-3(P3)         ;GET VARIABLE INDEX
04D1 01             XAE                     ;PUT IN E REG
04D2 C301           LD      1(P3)
04D4 CA80           ST      EREG(P2)        ;STORE LOWER 8 BITS
04D6 02             CCL                     ;  INTO VARIABLE
04D7 40             LDE                     ;INCREMENT INDEX
04D8 F401           ADI     1
04DA 01             XAE
04DB C302           LD      2(P3)
04DD CA80           ST      EREG(P2)        ;STORE UPPER 8 BITS
04DF 33             XPAL    P3              ;  INTO VARIABLE
04E0 CAFD           ST      LSTK(P2)        ;UPDATE STACK POINTER
04E2 C400   X10:    JS      P3,EXECIL


                    ;*************************************
                    ;*      TEST FOR VARIABLE IN TEXT    *
                    ;*************************************

04E9 C501   TSTVAR: LD      @1(P1)
04EB E420           XRI     ' '             ;SLEW OFF SPACES
04ED 98FA           JZ      TSTVAR
04EF C1FF           LD      -1(P1)          ;GET CHARACTER IN QUESTION
04F1 03             SCL
04F2 FC5B           CAI     'Z'+1           ;SUBTRACT 'Z'+1
04F4 9405           JP      $FAIL           ;NOT VARIABLE IF POSITIVE
04F6 03             SCL
04F7 FCE6           CAI     'A'-'Z'-1       ;SUBTRACT 'A'
04F9 9412           JP      $MAYBE          ;IF POS, MAY BE VARIABLE
04FB C5FF   $FAIL:  LD      @-1(P1)         ;BACKSPACE CURSOR
04FD C2FB           LD      PCLOW(P2)       ;GET TEST-FAIL ADDRESS
04FF 33             XPAL    P3              ;  FROM I.L. TABLE, PUT IT
0500 C2FA           LD      PCHIGH(P2)      ;  INTO I.L. PROGRAM COUNTER
0502 37             XPAH    P3
0503 C300           LD      (P3)
0505 CAFA           ST      PCHIGH(P2)
0507 C301           LD      1(P3)
0509 CAFB           ST      PCLOW(P2)
050B 90D5           JMP     X10
050D 01     $MAYBE: XAE                     ;SAVE VALUE (0-25)
050E C100           LD      (P1)            ;CHECK FOLLOWING CHAR
0510 03             SCL                     ;MUST NOT BE A LETTER
0511 FC5B           CAI     'Z'+1           ;  OTHERWISE WE'D BE LOOKING
0513 9405           JP      $OK             ;  AT A KEYWORD, NOT VARIABLE
0515 03             SCL
0516 FCE6           CAI     'A'-'Z'-1
0518 94E1           JP      $FAIL
051A C410   $OK:    LDI     H(AESTK)        ;SET P3 TO CURRENT
051C 37             XPAH    P3              ;  STACK LOCATION
051D AAFD           ILD     LSTK(P2)        ;INCR STACK POINTER
051F 33             XPAL    P3
0520 02             CCL                     ;DOUBLE VARIABLE INDEX
0521 40             LDE
0522 70             ADE
0523 CBFF           ST      -1(P3)          ;PUT INDEX ON STACK
0525 C402           LDI     2               ;INCREMENT I.L. PC, SKIPPING
0527 02             CCL                     ;  OVER TEST-FAIL ADDRESS
0528 F2FB           ADD     PCLOW(P2)
052A CAFB           ST      PCLOW(P2)
052C C400           LDI     0
052E F2FA           ADD     PCHIGH(P2)
0530 CAFA           ST      PCHIGH(P2)
0532 90AE           JMP     X10


                    ;*************************************
                    ;*     IND -- EVALUATE A VARIABLE    *
                    ;*************************************

0534 C410   IND:    LDI     H(AESTK)        ;SET P3 TO STACK
0536 37             XPAH    P3
0537 AAFD           ILD     LSTK(P2)
0539 33             XPAL    P3
053A C3FE           LD      -2(P3)          ;GET INDEX OFF TOP
053C 01             XAE                     ;PUT INDEX IN E REG
053D C280           LD      EREG(P2)        ;GET LOWER 8 BITS
053F CBFE           ST      -2(P3)          ;SAVE ON STACK
0541 02             CCL
0542 40             LDE                     ;INCREMENT E REG
0543 F401           ADI     1
0545 01             XAE
0546 C280           LD      EREG(P2)        ;GET UPPER 8 BITS
0548 CBFF           ST      -1(P3)          ;SAVE ON STACK
054A 9096   X11:    JMP     X10


                    ;*************************************
                    ;*     RELATIONAL OPERATORS          *
                    ;*************************************

054C C401   EQ:     LDI     1               ;EACH RELATIONAL OPERATOR
054E 9012           JMP     CMP             ;  LOADS A NUMBER USED LATER
0550 C402   NEQ:    LDI     2               ;  AS A CASE SELECTOR, AFTER
0552 900E           JMP     CMP             ;  THE TWO OPERANDS ARE COM-
0554 C403   LSS:    LDI     3               ;  PARED.  BASED ON THE COM-
0556 900A           JMP     CMP             ;  PARISON, FLAGS ARE SET THAT
0558 C404   LEQ:    LDI     4               ;  ARE EQUIVALENT TO THOSE SET
055A 9006           JMP     CMP             ;  BY THE 'CMP' INSTRUCTION IN
055C C405   GTR:    LDI     5               ;  THE PDP-11.  THESE PSEUDO-
055E 9002           JMP     CMP             ;  FLAGS ARE USED TO DETERMINE
0560 C406   GEQ:    LDI     6               ;  WHETHER THE PARTICULAR
                                            ;  RELATION IS SATISFIED OR NO
0562 CAEB   CMP:    ST      NUM(P2)
0564 C410           LDI     H(AESTK)        ;SET P3 -> ARITH STACK
0566 37             XPAH    P3
0567 BAFD           DLD     LSTK(P2)
0569 BAFD           DLD     LSTK(P2)
056B 33             XPAL    P3
056C 03             SCL
056D C3FE           LD      -2(P3)          ;SUBTRACT THE TWO OPERANDS,
056F FB00           CAD     (P3)            ;  STORING RESULT IN LO & HI
0571 CAEF           ST      LO(P2)
0573 C3FF           LD      -1(P3)
0575 FB01           CAD     1(P3)
0577 CAEE           ST      HI(P2)
0579 E3FF           XOR     -1(P3)          ;OVERFLOW OCCURS IF SIGNS OF
057B 01             XAE                     ;  RESULT AND 1ST OPERAND
057C C3FF           LD      -1(P3)          ;  DIFFER, AND SIGNS OF THE
057E E301           XOR     1(P3)           ;  TWO OPERANDS DIFFER
0580 50             ANE                     ;BIT 7 EQUIVALENT TO V FLAG
0581 E2EE           XOR     HI(P2)          ;BIT 7 EQUIVALENT TO N XOR V
0583 CAEA           ST      TEMP(P2)        ;STORE IN TEMP
0585 C2EE           LD      HI(P2)          ;DETERMINE IF RESULT WAS ZERO
0587 DAEF           OR      LO(P2)
0589 9802           JZ      SETZ            ;IF RESULT=0, SET Z FLAG
058B C480           LDI     080             ;  ELSE CLEAR Z FLAG
058D E480   SETZ:   XRI     080
058F 01             XAE                     ;BIT 7 OF EX = Z FLAG
0590 BAEB           DLD     NUM(P2)         ;TEST FOR =
0592 9C05           JNZ     NEQ1
0594 40             LDE                     ;  EQUAL IF Z = 1
0595 902B           JMP     CMP1
0597 90B1   X12:    JMP     X11
0599 BAEB   NEQ1:   DLD     NUM(P2)         ;TEST FOR <>
059B 9C05           JNZ     LSS1
059D 40             LDE                     ;  NOT EQUAL IF Z = 0
059E E480           XRI     080
05A0 9020           JMP     CMP1
05A2 BAEB   LSS1:   DLD     NUM(P2)         ;TEST FOR <
05A4 9C04           JNZ     LEQ1
05A6 C2EA           LD      TEMP(P2)        ;  LESS THAN IF (N XOR V)=1
05A8 9018           JMP     CMP1
05AA BAEB   LEQ1:   DLD     NUM(P2)         ;TEST FOR <=
05AC 9C05           JNZ     GTR1
05AE 40             LDE                     ;  LESS THAN OR EQUAL
05AF DAEA           OR      TEMP(P2)        ;   IF (Z OR (N XOR V))=1
05B1 900F           JMP     CMP1
05B3 BAEB   GTR1:   DLD     NUM(P2)         ;TEST FOR >
05B5 9C07           JNZ     GE01
05B7 40             LDE                     ;  GREATER THAN
05B8 DAEA           OR      TEMP(P2)        ;   IF (Z OR (N XOR V))=0
05BA E480           XRI     080
05BC 9004           JMP     CMP1
05BE C2EA   GE01:   LD      TEMP(P2)        ;GREATER THAN OR EQUAL
05C0 E480           XRI     080             ;  IF (N XOR V)=0
05C2 9404   CMP1:   JP      FALSE           ;IS RELATION SATISFIED?
05C4 C401           LDI     1               ;YES - PUSH 1 ON STACK
05C6 9002           JMP     CMP2
05C8 C400   FALSE:  LDI     0               ;NO - PUSH 0 ON STACK
05CA CBFE   CMP2:   ST      -2(P3)
05CC C400           LDI     0
05CE CBFF           ST      -1(P3)
05D0 C400           JS      P3,RTN          ;DO AN I.L. RETURN
05D2 90BE           JMP     X12


                    ;*************************************
                    ;*   IF STATEMENT TEST FOR ZERO      *
                    ;*************************************

05D4 C2EF   CMPR:   LD      LO(P2)          ;GET LOW & HI BYTES OF EXPR.
05D6 DAEE           OR      HI(P2)          ;TEST IF EXPRESSION IS ZERO
05D8 9802           JZ      FAIL            ;YES - IT IS
05DA 90B6           JMP     X12             ;NO - IT ISN'T SO CONTINUE
05DC C501   FAIL:   LD      @1(P1)          ;SKIP TO NEXT LINE IN PROGRAM
05DE E40D           XRI     0D              ;  (I.E. TIL NEXT CR)
05E0 9CFA           JNZ     FAIL
05E2 C402           JS      P3,NXT          ;CALL NXT AND RETURN
05E4 90A7   X12A:   JMP     X12


                    ;*************************************
                    ;*        AND, OR, & NOT             *
```

```
1047                    ;*************************************
1048
1049                            .LOCAL
1050 05F0 C401  ANDOP:  LDI     1               ;EACH OPERATION HAS ITS
1051 05F2 9006          JMP     $1              ; OWN CASE SELECTOR.
1052 05F4 C402  OROP:   LDI     2
1053 05F6 9002          JMP     $1
1054 05F8 C403  NOTOP:  LDI     3
1055 05FA CAEB  $1:     ST      NUM(P2)
1056 05FC C410          LDI     H(AESTK)        ;SET P3 -> ARITH. STACK
1057 05FE 37            XPAH    P3
1058 05FF BAFD          DLD     LSTK(P2)
1059 0601 BAFD          DLD     LSTK(P2)
1060 0603 33            XPAL    P3
1061 0604 BAEB          DLD     NUM(P2)         ;TEST FOR 'AND'
1062 0606 9C0E          JNZ     $OR
1063 0608 C301          LD      1(P3)           ;REPLACE TWO TOP ITEMS ON
1064 060A D3FF          AND     -1(P3)          ; STACK BY THEIR 'AND'
1065 060C CBFF          ST      -1(P3)
1066 060E C300          LD      0(P3)
1067 0610 D3FE          AND     -2(P3)
1068 0612 CBFE          ST      -2(P3)
1069 0614 90D8          JMP     X12A
1070 0616 BAEB  $OR:    DLD     NUM(P2)         ;TEST FOR 'OR'
1071 0618 9C0E          JNZ     $NOT
1072 061A C301          LD      1(P3)           ;REPLACE TWO TOP ITEMS ON
1073 061C DBFF          OR      -1(P3)          ; STACK BY THEIR 'OR'
1074 061E CBFF          ST      -1(P3)
1075 0620 C300          LD      0(P3)
1076 0622 DBFE          OR      -2(P3)
1077 0624 CBFE          ST      -2(P3)
1078 0626 90C6          JMP     X12A
1079 0628 C701  $NOT    LD      @1(P3)          ;'NOT' OPERATION
1080 062A E4FF          XRI     0FF
1081 062C CBFF          ST      -1(P3)          ;REPLACE TOP ITEM ON STACK
1082 062E C701          LD      @1(P3)          ; BY ITS ONE'S COMPLEMENT
1083 0630 E4FF          XRI     0FF
1084 0632 CBFF          ST      -1(P3)
1085 0634 33            XPAL    P3
1086 0635 CAFD          ST      LSTK(P2)        ;STACK POINTER FIXUP
1087 0637 90B5  X12B:   JMP     X12A
1088
1089
1090                    ;*************************************
1091                    ;*     EXCHANGE CURSOR WITH RAM      *
1092                    ;*************************************
1093
1094 0639 C2F1  XCHGP1: LD      P1LOW(P2)       ;THIS ROUTINE IS HANDY WHEN
1095 063B 31            XPAL    P1              ; EXECUTING AN 'INPUT' STMT
1096 063C CAF1          ST      P1LOW(P2)       ; IT EXCHANGES THE CURRENT
1097 063E C2F0          LD      P1HIGH(P2)      ; TEXT CURSOR WITH ONE SAVED
1098 0640 35            XPAH    P1              ; IN RAM
1099 0641 CAF0          ST      P1HIGH(P2)
1100 0643 3F            XPPC    P3
1101
1102
1103                    ;*************************************
1104                    ;*          CHECK RUN MODE           *
1105                    ;*************************************
1106
1107 0644 C2F4  CKMODE: LD      RUNMOD(P2)      ;THIS ROUTINE CAUSES AN ERROR
1108 0646 9801          JZ      CK1             ; IF CURRENTLY IN EDIT MODE
1109 0648 3F            XPPC    P3
1110 0649 C403  CK1:    LDI     3
1111 064B CAEB  E8:     ST      NUM(P2)         ;ERROR IF RUN MODE = 0
1112 064D C402          JS      P3,ERR2         ;MINOR KLUGE
1113
1114
1115                    ;*************************************
1116                    ;*      GET HEXADECIMAL NUMBER       *
1117                    ;*************************************
1118
1119                            LOCAL
1120 0654 AAFD  HEX:    ILD     LSTK(P2)        ;POINT P3 AT ARITH STACK
1121 0656 AAFD          ILD     LSTK(P2)
1122 0658 33            XPAL    P3
1123 0659 C410          LDI     H(AESTK)
1124 065B 37            XPAH    P3
1125 065C C400          LDI     0               ;NUMBER INITIALLY ZERO
1126 065E CBFF          ST      -1(P3)          ;PUT IT ON STACK
1127 0660 CBFE          ST      -2(P3)
1128 0662 CAEB          ST      NUM(P2)         ;ZERO NUMBER OF DIGITS
1129 0664 C501  $SKIP:  LD      @1(P1)          ;SKIP ANY SPACES
1130 0666 E420          XRI     ' '
1131 0668 98FA          JZ      $SKIP
1132 066A C5FF          LD      @-1(P1)
1133 066C C100  $LOOP:  LD      (P1)            ;GET A CHARACTER
1134 066E 03            SCL
1135 066F FC3A          CAI     '9'+1           ;CHECK FOR A NUMERIC CHAR
1136 0671 9409          JP      $LETR
1137 0673 03            SCL
1138 0674 FCF6          CAI     '0'-'9'-1       ;IF NUMERIC, SHIFT NUMBER
1139 0676 9413          JP      $ENTER          ; AND ADD NEW HEX DIGIT
1140 0678 9032          JMP     $END
1141 067A 90BB  X12C:   JMP     X12B
1142 067C 03    $LETR:  SCL                     ;CHECK FOR HEX LETTER
1143 067D FC0D          CAI     'G'-'9'-1
1144 067F 942B          JP      $END
1145 0681 03            SCL
1146 0682 FCFA          CAI     'A'-'G'
1147 0684 9402          JP      $OK
1148 0686 9024          JMP     $END
1149 0688 02    $OK:    CCL                     ;ADD 10 TO GET TRUE VALUE
1150 0689 F40A          ADI     10              ; OF LETTER
1151 068B 01    $ENTER: XAE                     ;NEW DIGIT IN EX REG
1152 068C C404          LDI     4               ;SET SHIFT COUNTER
1153 068E CAEA          ST      TEMP(P2)
1154 0690 CAEB          ST      NUM(P2)         ;DIGIT COUNT IS NON-ZERO
1155 0692 C3FE  $SHIFT: LD      -2(P3)          ;SHIFT NUMBER LEFT BY 4
1156 0694 02            CCL
1157 0695 F3FE          ADD     -2(P3)
1158 0697 CBFE          ST      -2(P3)
1159 0699 C3FF          LD      -1(P3)
1160 069B F3FF          ADD     -1(P3)
1161 069D CBFF          ST      -1(P3)
1162 069F BAEA          DLD     TEMP(P2)
1163 06A1 9CEF          JNZ     $SHIFT
1164 06A3 C3FE          LD      -2(P3)          ;ADD NEW DIGIT
1165 06A5 58            ORE                     ; INTO NUMBER
1166 06A6 CBFE          ST      -2(P3)
1167 06A8 C501          LD      @1(P1)          ;ADVANCE THE CURSOR
1168 06AA 90C0          JMP     $LOOP           ;GET NEXT CHAR
1169 06AC C2EB  $END:   LD      NUM(P2)         ;CHECK IF THERE WERE
1170 06AE 9C87          JNZ     X12B            ; MORE THAN 0 CHARACTERS
1171 06B0 C405          LDI     5               ;ERROR IF THERE WERE NONE
1172 06B2 9097  E8B:    JMP     E8
1173
1174
1175                    ;*************************************
1176                    ;*     TEST FOR NUMBER IN TEXT       *
1177                    ;*************************************
1178
1179                    ; THIS ROUTINE TESTS FOR A NUMBER IN THE TEXT.  IF NO
1180                    ; NUMBER IS FOUND, I.L. CONTROL PASSES TO THE ADDRESS
1181                    ; INDICATED IN THE 'TSTN' INSTRUCTION.  OTHERWISE, THE
1182                    ; NUMBER IS SCANNED AND PUT ON THE ARITHMETIC STACK,
1183                    ; WITH I.L. CONTROL PASSING TO THE NEXT INSTRUCTION.
1184
1185                            .LOCAL
1186 06B4 C501  TSTNUM: LD      @1(P1)
1187 06B6 E420          XRI     ' '             ;SKIP OVER ANY SPACES
1188 06B8 98FA          JZ      TSTNUM
1189 06BA C5FF          LD      @-1(P1)         ;GET FIRST CHAR
1190 06BC 03            SCL                     ;TEST FOR DIGIT
1191 06BD FC3A          CAI     '9'+1
1192 06BF 9405          JP      $ABORT
1193 06C1 03            SCL
1194 06C2 FCF6          CAI     '0'-'9'-1
1195 06C4 9421          JP      $1
1196 06C6 C2FB  $ABORT: LD      PCLOW(P2)       ;GET TEST-FAIL ADDRESS
1197 06C8 33            XPAL    P3              ; FROM I.L. TABLE
1198 06C9 C2FA          LD      PCHIGH(P2)
1199 06CB 37            XPAH    P3
1200 06CC C300          LD      (P3)            ;PUT TEST-FAIL ADDRESS
1201 06CE CAFA          ST      PCHIGH(P2)      ; INTO I.L. PC
1202 06D0 C301          LD      1(P3)
1203 06D2 CAFB          ST      PCLOW(P2)
1204 06D4 90A4          JMP     X12C
1205 06D6 C402  $RET:   LDI     2               ;SKIP OVER ONE IL INSTRUCTION
1206 06D8 02            CCL                     ; IF NUMBER IS DONE
1207 06D9 F2FB          ADD     PCLOW(P2)
1208 06DB CAFB          ST      PCLOW(P2)
1209 06DD C400          LDI     0
1210 06DF F2FA          ADD     PCHIGH(P2)
1211 06E1 CAFA          ST      PCHIGH(P2)
1212 06E3 9095  X13:    JMP     X12C
1213 06E5 90CB  E8A:    JMP     E8B
1214 06E7 01    $1:     XAE                     ;SAVE DIGIT IN EX REG
1215 06E8 C410          LDI     H(AESTK)        ;POINT P3 AT AE STACK
1216 06EA 37            XPAH    P3
1217 06EB AAFD          ILD     LSTK(P2)
1218 06ED AAFD          ILD     LSTK(P2)
1219 06EF 33            XPAL    P3
1220 06F0 C400          LDI     0
1221 06F2 CBFF          ST      -1(P3)
1222 06F4 40            LDE
1223 06F5 CBFE          ST      -2(P3)
1224 06F7 C501  $LOOP:  LD      @1(P1)          ;GET NEXT CHAR
1225 06F9 C100          LD      (P1)
1226 06FB 03            SCL                     ;TEST IF IT IS DIGIT
1227 06FC FC3A          CAI     '9'+1
1228 06FE 94D6          JP      $RET            ;RETURN IF IT ISN'T
1229 0700 03            SCL
1230 0701 FCF6          CAI     '0'-'9'-1
1231 0703 9402          JP      $2
1232 0705 90CF          JMP     $RET
1233 0707 01    $2:     XAE                     ;SAVE DIGIT
1234 0708 C3FF          LD      -1(P3)          ;PUT RESULT IN SCRATCH SPACE
1235 070A CB01          ST      1(P3)
1236 070C C3FE          LD      -2(P3)
1237 070E CB00          ST      (P3)
1238 0710 C402          LDI     2
1239 0712 CAEA          ST      TEMP(P2)        ;MULTIPLY RESULT BY 10
1240 0714 02    $SHIFT: CCL                     ;FIRST MULTIPLY BY 4
1241 0715 C3FE          LD      -2(P3)
1242 0717 F3FE          ADD     -2(P3)
1243 0719 CBFE          ST      -2(P3)
1244 071B C3FF          LD      -1(P3)
1245 071D F3FF          ADD     -1(P3)
1246 071F CBFF          ST      -1(P3)
1247 0721 BAEA          DLD     TEMP(P2)
1248 0723 9CEF          JNZ     $SHIFT
1249 0725 02            CCL                     ;THEN ADD OLD RESULT,
1250 0726 C3FE          LD      -2(P3)          ; SO WE HAVE RESULT * 5
1251 0728 F300          ADD     (P3)
1252 072A CBFE          ST      -2(P3)
1253 072C C3FF          LD      -1(P3)
1254 072E F301          ADD     1(P3)
1255 0730 CBFF          ST      -1(P3)
1256 0732 02            CCL                     ;THEN MULTIPLY BY TWO
1257 0733 C3FE          LD      -2(P3)
1258 0735 F3FE          ADD     -2(P3)
1259 0737 CBFE          ST      -2(P3)
1260 0739 C3FF          LD      -1(P3)
1261 073B F3FF          ADD     -1(P3)
1262 073D CBFF          ST      -1(P3)
1263 073F 02            CCL                     ;THEN ADD IN NEW DIGIT
1264 0740 40            LDE
1265 0741 F3FE          ADD     -2(P3)
1266 0743 CBFE          ST      -2(P3)
1267 0745 C400          LDI     0
1268 0747 F3FF          ADD     -1(P3)
1269 0749 CBFF          ST      -1(P3)
1270 074B 94AA          JP      $LOOP           ;REPEAT IF NO OVERFLOW
1271 074D C406          LDI     6
1272 074F 9094  E9:     JMP     E8A             ;ELSE REPORT ERROR
1273 0751 9090  X14:    JMP     X13
1274
1275
1276                    ;*************************************
1277                    ;*    GET LINE FROM TELETYPE         *
1278                    ;*************************************
1279
1280                            LOCAL
1281 0753       GETL:   LDPI    P1,LBUF         ;SET P1 TO LBUF
1282 0759 C400          LDI     0               ;CLEAR NO. OF CHAR
1283 075B CAE7          ST      CHRNUM(P2)
1284 075D               LDPI    P3,PUTC-1       ;POINT P3 AT PUTC ROUTINE
1285 0763 C2F4          LD      RUNMOD(P2)      ;PRINT '? ' IF RUNNING
1286 0765 9808          JZ      $0              ; (I.E. DURING 'INPUT')
1287 0767 C43F          LDI     '?'
1288 0769 3F            XPPC    P3
1289 076A C420          LDI     ' '
1290 076C 3F            XPPC    P3
1291 076D 9003          JMP     $1
1292 076F C43E  $0:     LDI     '>'             ;OTHERWISE PRINT '>'
1293 0771 3F            XPPC    P3
1294 0772 C40F  $1:     JS      P3,GECO         ;GET CHARACTER
1295 0779 C4BD          LDI     L(PUTC)-1       ;POINT P3 AT PUTC AGAIN
1296 077B 33            XPAL    P3
1297 077C 40            LDE                     ;GET TYPED CHAR
1298 077D 98F3          JZ      $1              ;IGNORE NULLS
1299 077F E40A          XRI     0A              ;IGNORE LINE FEED
1300 0781 98EF          JZ      $1
1301 0783 40            LDE
1302 0784 E40D          XRI     0D              ;CHECK FOR CR
1303 0786 9850          JZ      $CR
1304 0788 40            LDE
1305 0789 E45F          XRI     'O'+010         ;CHECK FOR SHIFT/O
1306 078B 9841          JZ      $RUB
1307 078D 40            LDE                     ;CHECK FOR CTRL/H
1308 078E E408          XRI     8
1309 0790 9836          JZ      $XH
1310 0792 40            LDE
1311 0793 E415          XRI     015             ;CHECK FOR CTRL/U
1312 0795 980F          JZ      $XU
1313 0797 40            LDE
1314 0798 E403          XRI     3               ;CHECK FOR CTRL/C
```

```
1315 079A 9C1A          JNZ     $ENTER
1316 079C C45E          LDI     '^'             ;ECHO CONTROL/C AS ^C
1317 079E 3F            XPPC    P3
1318 079F C443          LDI     'C'
1319 07A1 3F            XPPC    P3
1320 07A2 C40E          LDI     14              ;CAUSE A BREAK
1321 07A4 90A9          JMP     E9
1322 07A6 C45E  $XU:    LDI     '^'             ;ECHO CONTROL/U AS ^U
1323 07A8 3F            XPPC    P3
1324 07A9 C455          LDI     'U'
1325 07AB 3F            XPPC    P3
1326 07AC C40D          LDI     0D              ;PRINT CR/LF
1327 07AE 3F            XPPC    P3
1328 07AF C40A          LDI     0A
1329 07B1 3F            XPPC    P3
1330 07B2 909F  $2:     JMP     GETL            ;GO GET ANOTHER LINE
1331 07B4 909B  X15:    JMP     X14
1332 07B6 40    $ENTER: LDE
1333 07B7 CD01          ST      @1(P1)          ;PUT CHAR IN LBUF
1334 07B9 AAE7          ILD     CHRNUM(P2)      ;INCREMENT CHRNUM
1335 07BB E448          XRI     72              ;IF=72, LINE FULL
1336 07BD 9CB3          JNZ     $1
1337 07BF C40D          LDI     0D
1338 07C1 01            XAE                     ;SAVE CARRIAGE RET
1339 07C2 40            LDE
1340 07C3 3F            XPPC    P3              ;PRINT IT
1341 07C4 9012          JMP     $CR             ;STORE IT IN LBUF
1342 07C6 9087  E10:    JMP     E9
1343 07C8 C420  $XH:    LDI     ' '             ;BLANK OUT THE CHARACTER
1344 07CA 3F            XPPC    P3
1345 07CB C408          LDI     8               ;PRINT ANOTHER BACKSPACE
1346 07CD 3F            XPPC    P3
1347 07CE C2E7  $RUB:   LD      CHRNUM(P2)
1348 07D0 98A0          JZ      $1
1349 07D2 BAE7          DLD     CHRNUM(P2)      ;ONE LESS CHAR
1350 07D4 C5FF          LD      @-1(P1)         ;BACKSPACE CURSOR
1351 07D6 909A          JMP     $1
1352 07D8 40    $CR:    LDE
1353 07D9 CD01          ST      @1(P1)          ;STORE CR IN LBUF
1354 07DB C40A          LDI     0A              ;PRINT LINE FEED
1355 07DD 3F            XPPC    P3
1356 07DE C410          LDI     H(LBUF)         ;SET P1 TO BEGIN-
1357 07E0 35            XPAH    P1              ; NING OF LBUF
1358 07E1 C4D6          LDI     L(LBUF)
1359 07E3 31            XPAL    P1
1360 07E4 90CE  X16:    JMP     X15
1361
1362
1363                    ;************************************
1364                    ;*      EVAL -- GET MEMORY CONTENTS   *
1365                    ;************************************
1366
1367                    ; THIS ROUTINE IMPLEMENTS THE '@' OPERATOR IN EXPRESSIONS
1368
1369 07E6 C410 EVAL:   LDI     H(AESTK)
1370 07E8 37            XPAH    P3
1371 07E9 C2FD          LD      LSTK(P2)
1372 07EB 33            XPAL    P3              ;P3 -> ARITH STACK
1373 07EC C3FF          LD      -1(P3)          ;GET ADDR OFF STACK,
1374 07EE 35            XPAH    P1              ; AND INTO P1,
1375 07EF 01            XAE                     ; SAVING OLD P1 IN EX & LO
1376 07F0 C3FE          LD      -2(P3)
1377 07F2 31            XPAL    P1
1378 07F3 CAEF          ST      LO(P2)
1379 07F5 C100          LD      0(P1)           ;GET MEMORY CONTENTS,
1380 07F7 CBFE          ST      -2(P3)          ; SHOVE ONTO STACK
1381 07F9 C400          LDI     0
1382 07FB CBFF          ST      -1(P3)          ;HIGH ORDER 8 BITS ZEROED
1383 07FD C2EF          LD      LO(P2)
1384 07FF 31            XPAL    P1              ;RESTORE ORIGINAL P1
1385 0800 40            LDE
1386 0801 35            XPAH    P1
1387 0802 90B0          JMP     X15
1388
1389
1390                    ;************************************
1391                    ;*    MOVE -- STORE INTO MEMORY     *
1392                    ;************************************
1393
1394                    ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
1395                    ;    '@' FACTOR '=' REL-EXP
1396
1397 0804 C410 MOVE:    LDI     H(AESTK)
1398 0806 37            XPAH    P3
1399 0807 C2FD          LD      LSTK(P2)
1400 0809 33            XPAL    P3              ;P3 -> ARITH STACK
1401 080A C7FE          LD      @-2(P3)         ;GET BYTE TO BE MOVED
1402 080C 01            XAE
1403 080D C7FF          LD      @-1(P3)         ;NOW GET ADDRESS INTO P3
1404 080F CAEA          ST      TEMP(P2)
1405 0811 C7FF          LD      @-1(P3)
1406 0813 33            XPAL    P3
1407 0814 CAFD          ST      LSTK(P2)        ;STACK PTR UPDATED NOW
1408 0816 C2EA          LD      TEMP(P2)
1409 0818 37            XPAH    P3
1410 0819 40            LDE
1411 081A CB00          ST      0(P3)           ;MOVE THE BYTE INTO MEMORY
1412 081C 90C6  X17:    JMP     X16
1413 081E 90A6  E11:    JMP     E10
1414
1415                    ;************************************
1416                    ;*          TEXT EDITOR             *
1417                    ;************************************
1418
1419                    ;INPUTS TO THIS ROUTINE: POINTER TO LINE BUFFER IN P1LOW &
1420                    ; P1HIGH.  P1 POINTS TO THE INSERTION POINT IN THE TEXT.
1421                    ; THE A.E. STACK HAS THE LINE NUMBER ON IT (STACK POINTER
1422                    ; IS ALREADY POPPED).
1423
1424                    ;EACH LINE IN THE NIBL TEXT IS STORED IN THE FOLLOWING
1425                    ; FORMAT:  TWO BYTES CONTAINING THE LINE NUMBER (IN BINARY,
1426                    ; HIGH ORDER BYTE FIRST), THEN ONE BYTE CONTAINING THE
1427                    ; LENGTH OF THE LINE, AND FINALLY THE LINE ITSELF FOLLOWED
1428                    ; BY A CARRIAGE RETURN.   THE LAST LINE IN THE TEXT IS
1429                    ; FOLLOWED BY TWO CONSECUTIVE BYTES OF X'FF.
1430
1431                            .LOCAL
1432 0820 C410 INSRT:   LDI     H(AESTK)        ;POINT P3 AT AE STACK,
1433 0822 37            XPAH    P3              ; WHICH HAS THE LINE #
1434 0823 C2FD          LD      LSTK(P2)        ; ON IT
1435 0825 33            XPAL    P3
1436 0826 C301          LD      1(P3)           ;SAVE NEW LINE'S NUMBER
1437 0828 CAF7          ST      HILINE(P2)
1438 082A C300          LD      0(P3)
1439 082C CAF8          ST      LOLINE(P2)
1440 082E C2F1          LD      P1LOW(P2)       ;PUT POINTER TO LBUF INTO P3
1441 0830 33            XPAL    P3
1442 0831 C2F0          LD      P1HIGH(P2)
1443 0833 37            XPAH    P3
1444 0834 C404          LDI     4               ;INITIALLY LENGTH OF NEW LINE
1445 0836 CAE7          ST      CHRNUM(P2)      ; = 4.  ADD 1 TO LENGTH FOR
1446 0838 C701  $1:     LD      @1(P3)          ; EACH CHAR IN LINE UP TO, BU
1447 083A E40D          XRI     0D              ; NOT INCLUDING, CARR. RETURN
1448 083C 9804          JZ      $2
1449 083E AAE7          ILD     CHRNUM(P2)
1450 0840 90F6          JMP     $1
1451 0842 C2E7  $2:     LD      CHRNUM(P2)      ;IF LENGTH STILL 4, WE'LL DEL
1452 0844 E404          XRI     4               ; A LINE, SO SET LENGTH = 0
1453 0846 9C02          JNZ     $3
1454 0848 CAE7          ST      CHRNUM(P2)
1455 084A C2E7  $3:     LD      CHRNUM(P2)      ;PUT NEW LINE LENGTH IN EX
1456 084C 01            XAE
1457 084D C2F2          LD      LABLHI(P2)      ;IS NEW LINE REPLACING OLD?
1458 084F 9406          JP      $4              ;YES - DO REPLACE
1459 0851 D47F          ANI     07F             ;NO - WE'LL INSERT LINE HERE,
1460 0853 CAF2          ST      LABLHI(P2)      ; WHERE FNDLBL GOT US.
1461 0855 9018          JMP     $MOVE           ;BUT FIRST MAKE ROOM
1462 0857 C503  $4:     LD      @3(P1)          ;SKIP LINE # AND LENGTH
1463 0859 40            LDE                     ;EX, NOW HOLDING NEW LINE
1464 085A 02            CCL                     ; LENGTH, WILL SOON HOLD
1465 085B F4FC          ADI     -4              ; DISPLACEMENT OF LINES
1466 085D 01            XAE                     ; TO BE MOVED
1467 085E C501  $5:     LD      @1(P1)          ;SUBTRACT 1 FROM DISPLACEMENT
1468 0860 E40D          XRI     0D              ; FOR EACH CHAR IN LINE BEING
1469 0862 980B          JZ      $MOVE           ; REPLACED
1470 0864 40            LDE
1471 0865 02            CCL
1472 0866 F4FF          ADI     -1
1473 0868 01            XAE
1474 0869 90F3          JMP     $5
1475 086B 90AF  X19:    JMP     X17
1476 086D 90AF  E12:    JMP     E11
1477 086F 40    $MOVE:  LDE                     ;IF DISPLACEMENT AND LENGTH
1478 0870 DAE7          OR      CHRNUM(P2)      ; OF NEW LINE ARE 0, RETURN
1479 0872 98F7          JZ      X19
1480 0874 C47A          LDI     L(DOSTAK)       ;CLEAR SOME STACKS
1481 0876 CAFF          ST      DOPTR(P2)
1482 0878 C46A          LDI     L(SBRSTK)
1483 087A CAFC          ST      SBRPTR(P2)
1484 087C C48A          LDI     L(FORSTK)
1485 087E CAFE          ST      FORPTR(P2)
1486 0880 40            LDE
1487 0881 9860          JZ      $ADD            ;DON'T NEED TO MOVE LINES
1488 0883 9410          JP      $UP             ;SKIP IF DISPLACEMENT POSITIV
1489 0885 C100  $DOWN:  LD      0(P1)           ;NEGATIVE DISPLACEMENT:
1490 0887 C980          ST      EREG(P1)        ;DO;
1491 0889 C501          LD      @1(P1)          ;    M(P1+DISP) = M(P1);
1492 088B 94F8          JP      $DOWN           ;    P1 = P1+1;
1493 088D C100          LD      0(P1)           ;UNTIL M(P1)<0 & M(P1-1)<0;
1494 088F 94F4          JP      $DOWN
1495 0891 C980          ST      EREG(P1)        ;M(P1+DISP) = M(P1);
1496 0893 904E          JMP     $ADD
1497 0895 C1FE  $UP:    LD      -2(P1)          ;POSITIVE DISPLACEMENT:
1498 0897 CAEA          ST      TEMP(P2)        ;FLAG BEGINNING OF MOVE WITH
1499 0899 C4FF          LDI     -1              ; A -1 FOLLOWED BY 80, WHICH
1500 089B C9FE          ST      -2(P1)          ; CAN NEVER APPEAR IN A
1501 089D C450          LDI     80              ; NIBL TEXT
1502 089F C9FF          ST      -1(P1)
1503 08A1 C501  $UP1:   LD      @1(P1)          ;ADVANCE P1 TO END OF TEXT
1504 08A3 94FC          JP      $UP1
1505 08A5 C100          LD      0(P1)
1506 08A7 94F8          JP      $UP1
1507 08A9 35            XPAH    P1              ;SAVE P1 IN LO, HI
1508 08AA CAEE          ST      HI(P2)
1509 08AC 35            XPAH    P1
1510 08AD 31            XPAL    P1
1511 08AE CAEF          ST      LO(P2)
1512 08B0 31            XPAL    P1

1513 08B1 C2EF          LD      LO(P2)          ;ADD DISPLACEMENT TO
1514 08B3 02            CCL                     ; VALUE OF P1, TO CHECK
1515 08B4 70            ADE                     ; WHETHER WE'RE OUT OF
1516 08B5 C400          LDI     0               ; RAM FOR USER'S PROGRAM
1517 08B7 F2EE          ADD     HI(P2)
1518 08B9 E2EE          XOR     HI(P2)
1519 08BB D4F0          ANI     0F0
1520 08BD 9803          JZ      $UP2
1521 08BF C400          LDI     0               ;IF OUT OF RAM, CHANGE
1522 08C1 01            XAE                     ; DISPLACEMENT TO ZERO
1523 08C2 C4FF  $UP2:   LDI     -1
1524 08C4 C980  $UP3:   ST      EREG(P1)        ;MOVE TEXT UP UNTIL WE REACH
1525 08C6 C5FF          LD      @-1(P1)         ; THE FLAGS SET ABOVE
1526 08C8 94FA          JP      $UP3
1527 08CA C101          LD      1(P1)
1528 08CC E450          XRI     80
1529 08CE 9804          JZ      $UP4
1530 08D0 C100          LD      0(P1)
1531 08D2 90F0          JMP     $UP3
1532 08D4 C2EA  $UP4:   LD      TEMP(P2)        ;RESTORE THE FLAGGED LOCATION
1533 08D6 C900          ST      0(P1)           ; TO THEIR ORIGINAL VALUES
1534 08D8 C40D          LDI     0D
1535 08DA C901          ST      1(P1)
1536 08DC 40            LDE
1537 08DD 9C04          JNZ     $ADD            ;IF DISPLACEMENT = 0, WE'RE
1538 08DF C402          LDI     2               ; OUT OF RAM, SO REPORT ERROR
1539 08E1 908A  E12A:   JMP     E12
1540 08E3 C2E7  $ADD:   LD      CHRNUM(P2)      ;INSERT NEW LINE
1541 08E5 9884  X19A:   JZ      X19             ; UNLESS LENGTH IS ZERO
1542 08E7 C2F1          LD      P1LOW(P2)       ;POINT P1 AT LINE BUFFER
1543 08E9 31            XPAL    P1
1544 08EA C2F0          LD      P1HIGH(P2)
1545 08EC 35            XPAH    P1
1546 08ED C2F3          LD      LABLLO(P2)      ;POINT P3 AT INSERTION PLACE
1547 08EF 33            XPAL    P3
1548 08F0 C2F2          LD      LABLHI(P2)
1549 08F2 37            XPAH    P3
1550 08F3 C2F7          LD      HILINE(P2)      ;PUT LINE NUMBER INTO TEXT
1551 08F5 CF01          ST      @1(P3)
1552 08F7 C2F8          LD      LOLINE(P2)
1553 08F9 CF01          ST      @1(P3)
1554 08FB C2E7          LD      CHRNUM(P2)      ;STORE LINE LENGTH IN TEXT
1555 08FD CF01          ST      @1(P3)
1556 08FF C501  $ADD1:  LD      @1(P1)          ;PUT REST OF CHARS
1557 0901 CF01          ST      @1(P3)          ; (INCLUDING CR) INTO TEXT
1558 0903 E40D          XRI     0D
1559 0905 9CF8          JNZ     $ADD1
1560 0907 90DC          JMP     X19A            ;RETURN
1561 0909 C400  X20:    JS      P3,EXECIL
1562 0910 90CF  E13:    JMP     E12A
1563
1564
1565                    ;************************************
1566                    ;*      POP ARITHMETIC STACK        *
1567                    ;************************************
1568
1569 0912 BAFD  POPAE:  DLD     LSTK(P2)        ;THIS ROUTINE POP THE A.E.
1570 0914 BAFD          DLD     LSTK(P2)        ; STACK, AND PUTS THE RESULT
1571 0916 33            XPAL    P3              ; INTO LO(P2) AND HI(P2)
1572 0917 C410          LDI     H(AESTK)
1573 0919 37            XPAH    P3
1574 091A C300          LD      (P3)
1575 091C CAEF          ST      LO(P2)
1576 091E C301          LD      1(P3)
1577 0920 CAEE          ST      HI(P2)
1578 0922 90E5          JMP     X20
1579
1580
```

```
1581                    ;*************************************
1582                    ;*              UNTIL                *
1583                    ;*************************************
1584
1585                            .LOCAL
1586 0924 C2FF  UNTIL:  LD      DOPTR(P2)       ;CHECK FOR DO-STACK UNDERFLOW
1587 0926 01            XAE
1588 0927 40            LDE
1589 0928 E47A          XRI     L(DOSTAK)
1590 092A 9C04          JNZ     $1
1591 092C C40F          LDI     15
1592 092E 90E0          JMP     E13
1593 0930 C2EF  $1:     LD      LO(P2)          ;CHECK FOR EXPRESSION = 0
1594 0932 DAEE          OR      HI(P2)
1595 0934 9806          JZ      $REDO           ;IF ZERO, REPEAT DO-LOOP
1596 0936 BAFF          DLD     DOPTR(P2)       ;ELSE POP SAVE STACK
1597 0938 BAFF          DLD     DOPTR(P2)
1598 093A 90CD          JMP     X20             ;CONTINUE TO NEXT STMT
1599 093C 40    $REDO:  LDE                     ;POINT P3 AT DO-STACK
1600 093D 33            XPAL    P3
1601 093E C410          LDI     H(DOSTAK)
1602 0940 37            XPAH    P3
1603 0941 C3FF          LD      -1(P3)          ;LOAD P1 FROM DO STACK
1604 0943 35            XPAH    P1
1605 0944 C3FE          LD      -2(P3)
1606 0946 31            XPAL    P1              ;CURSOR NOW POINTS TO FIRST
1607 0947 90C0          JMP     X20             ; STATEMENT OF DO-LOOP
1608
1609
1610                    ;*************************************
1611                    ;*     STORE INTO STATUS REGISTER    *
1612                    ;*************************************
1613
1614                    ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
1615                    ;       'STAT' '=' REL-EXP
1616
1617 0949 C2EF  MOVESR: LD      LO(P2)          ;LOW BYTE GOES TO STATUS
1618 094B D4F7          ANI     0F7             ; BUT WITH IEN BIT CLEARED
1619 094D 07            CAS
1620 094E 90B9  X21:    JMP     X20
1621 0950 90BE  E14:    JMP     E13
1622
1623
1624                    ;*************************************
1625                    ;*         STAT FUNCTION             *
1626                    ;*************************************
1627
1628 0952 C410  STATUS: LDI     H(AESTK)
1629 0954 37            XPAH    P3              ;POINT P3 AT AE STACK
1630 0955 AAFD          ILD     LSTK(P2)
1631 0957 AAFD          ILD     LSTK(P2)
1632 0959 33            XPAL    P3
1633 095A 06            CSA
1634 095B CBFE          ST      -2(P3)          ;STATUS REG IS LOW BYTE
1635 095D C400          LDI     0
1636 095F CBFF          ST      -1(P3)          ;ZERO IS HIGH BYTE
1637 0961 90EB          JMP     X21
1638
1639
1640                    ;*************************************
1641                    ;*   MACHINE LANGUAGE SUBROUTINE     *
1642                    ;*************************************
1643
1644                    ; THIS ROUTINE IMPLEMENTS THE 'LINK' STATEMENT
1645
1646 0963 C2EE  CALLML: LD      HI(P2)          ;GET HIGH BYTE OF ADDRESS
1647 0965 37            XPAH    P3
1648 0966 C2EF          LD      LO(P2)          ;GET LOW BYTE
1649 0968 33            XPAL    P3              ;P3 -> USER'S ROUTINE
1650 0969 C7FF          LD      @-1(P3)         ;CORRECT P3
1651 096B 3F            XPPC    P3              ;CALL ROUTINE (PRAY IT WORKS)
1652 096C               LDPI    P2,VARS         ;RESTORE RAM POINTER
1653 0972 90DA          JMP     X21             ;RETURN
1654
1655
1656                    ;*************************************
1657                    ;*       SAVE DO LOOP ADDRESS        *
1658                    ;*************************************
1659
1660                    ; THIS ROUTINE IMPLEMENTS THE 'DO' STATEMENT.
1661
1662                            .LOCAL
1663 0974 C2FF  SAVEDO: LD      DOPTR(P2)       ;CHECK FOR STACK OVERFLOW
1664 0976 E48A          XRI     L(FORSTK)
1665 0978 9C04          JNZ     $1
1666 097A C40A          LDI     10
1667 097C 90D2  E15:    JMP     E14
1668 097E AAFF  $1:     ILD     DOPTR(P2)
1669 0980 AAFF          ILD     DOPTR(P2)
1670 0982 33            XPAL    P3
1671 0983 C410          LDI     H(DOSTAK)
1672 0985 37            XPAH    P3              ;P3 -> TOP OF DO STACK
1673 0986 35            XPAH    P1              ;SAVE CURSOR ON THE STACK
1674 0987 CBFF          ST      -1(P3)
1675 0989 35            XPAH    P1
1676 098A 31            XPAL    P1
1677 098B CBFE          ST      -2(P3)
1678 098D 31            XPAL    P1
1679 098E 90BE  X22:    JMP     X21
1680
1681
1682                    ;*************************************
1683                    ;*        TOP OF RAM FUNCTION        *
1684                    ;*************************************
1685
1686                            .LOCAL
1687 0990 C2E9  TOP:    LD      TEMP2(P2)       ;SET P3 TO POINT TO
1688 0992 37            XPAH    P3              ; START OF NIBL TEXT
1689 0993 C2E8          LD      TEMP3(P2)
1690 0995 33            XPAL    P3
1691 0996 C300  $0:     LD      (P3)            ;HAVE WE HIT END OF TEXT?
1692 0998 9402          JP      $1              ;NO - SKIP TO NEXT LINE
1693 099A 9007          JMP     $2              ;YES - PUT CURSOR ON STACK
1694 099C C302  $1:     LD      2(P3)           ;GET LENGTH OF LINE
1695 099E 01            XAE
1696 099F C780          LD      @EREG(P3)       ;SKIP TO NEXT LINE
1697 09A1 90F3          JMP     $0              ;GO CHECK FOR EOF
1698 09A3 C702  $2:     LD      @2(P3)          ;P3 := P3 + 2
1699 09A5 AAFD          ILD     LSTK(P2)        ;SET P3 TO STACK, SAVING
1700 09A7 AAFD          ILD     LSTK(P2)        ; OLD P3 (WHICH CONTAINS TOP)
1701 09A9 33            XPAL    P3              ; ON IT SOMEHOW
1702 09AA 01            XAE
1703 09AB C410          LDI     H(AESTK)
1704 09AD 37            XPAH    P3
1705 09AE CBFF          ST      -1(P3)
1706 09B0 40            LDE
1707 09B1 CBFE          ST      -2(P3)
1708 09B3 90D9          JMP     X22
1709
1710
1711                    ;*************************************
1712                    ;*       SKIP TO NEXT NIBL LINE      *
1713                    ;*************************************
1714
1715 09B5 C501  IGNORE: LD      @1(P1)          ;SCAN TIL WE'RE PAST
1716 09B7 E40D          XRI     0D              ; CARRIAGE RETURN
1717 09B9 9CFA          JNZ     IGNORE
1718 09BB 3F            XPPC    P3              ;YES - RETURN
1719
1720
1721                    ;*************************************
1722                    ;*         MODULO FUNCTION           *
1723                    ;*************************************
1724
1725 09BC C2FD  MODULO: LD      LSTK(P2)        ;THIS ROUTINE MUST BE
1726 09BE 33            XPAL    P3              ; IMMEDIATELY AFTER A
1727 09BF C410          LDI     H(AESTK)        ; DIVIDE TO WORK CORRECTLY
1728 09C1 37            XPAH    P3
1729 09C2 C303          LD      3(P3)           ;GET LOW BYTE OF REMAINDER
1730 09C4 CBFE          ST      -2(P3)          ;PUT ON STACK
1731 09C6 C302          LD      2(P3)           ;GET HIGH BYTE OF REMAINDER
1732 09C8 CBFF          ST      -1(P3)          ;PUT ON STACK
1733 09CA 90C2  X23:    JMP     X22
1734 09CC 90AE  E16:    JMP     E15
1735
1736
1737                    ;*************************************
1738                    ;*          RANDOM FUNCTION          *
1739                    ;*************************************
1740
1741                            .LOCAL
1742 09CE C408  RANDOM: LDI     8               ;LOOP COUNTER FOR MULTIPLY
1743 09D0 CAEB          ST      NUM(P2)
1744 09D2 C2E5          LD      RNDX(P2)
1745 09D4 01            XAE
1746 09D5 C2E4          LD      RNDY(P2)
1747 09D7 CAE9          ST      TEMP2(P2)
1748 09D9 C2E5  $LOOP:  LD      RNDX(P2)        ;MULTIPLY THE SEEDS BY 9
1749 09DB 02            CCL
1750 09DC 70            ADE
1751 09DD 01            XAE
1752 09DE C2E4          LD      RNDY(P2)
1753 09E0 02            CCL
1754 09E1 F2E9          ADD     TEMP2(P2)
1755 09E3 CAE4          ST      RNDY(P2)
1756 09E5 BAEB          DLD     NUM(P2)
1757 09E7 9CF0          JNZ     $LOOP
1758 09E9 40            LDE                     ;ADD 7 TO SEEDS
1759 09EA 02            CCL
1760 09EB F407          ADI     7
1761 09ED 01            XAE
1762 09EE C2E4          LD      RNDY(P2)
1763 09F0 02            CCL
1764 09F1 F407          ADI     7
1765 09F3 1E            RR
1766 09F4 CAE4          ST      RNDY(P2)
1767 09F6 AAE6          ILD     RNDF(P2)        ;HAVE WE GONE THROUGH
1768 09F8 9803          JZ      $1              ; 256 GENERATIONS?
1769 09FA 40            LDE                     ;IF SO, SKIP GENERATING
1770 09FB CAE5          ST      RNDX(P2)        ; THE NEW RNDX
1771 09FD C2FD  $1:     LD      LSTK(P2)        ;START MESSING WITH THE STACK
1772 09FF 33            XPAL    P3
1773 0A00 C410          LDI     H(AESTK)
1774 0A02 37            XPAH    P3
1775 0A03 C401          LDI     1               ;FIRST PUT 1 ON STACK
1776 0A05 CB00          ST      (P3)
1777 0A07 C400          LDI     0
1778 0A09 CB01          ST      1(P3)
1779 0A0B C3FE          LD      -2(P3)          ;PUT EXPR2 ON STACK
1780 0A0D CB02          ST      2(P3)
1781 0A0F C3FF          LD      -1(P3)
1782 0A11 CB03          ST      3(P3)
1783 0A13 C3FC          LD      -4(P3)          ;PUT EXPR1 ON STACK
1784 0A15 CB04          ST      4(P3)
1785 0A17 C3FD          LD      -3(P3)
1786 0A19 CB05          ST      5(P3)
1787 0A1B C2E4          LD      RNDY(P2)        ;PUT RANDOM # ON STACK
1788 0A1D CBFE          ST      -2(P3)
1789 0A1F C2E5          LD      RNDX(P2)
1790 0A21 E4FF          XRI     0FF
1791 0A23 D47F          ANI     07F
1792 0A25 CBFF          ST      -1(P3)
1793 0A27 C706          LD      @6(P3)          ;ADD 6 TO STACK POINTER
1794 0A29 33            XPAL    P3
1795 0A2A CAFD          ST      LSTK(P2)
1796 0A2C 909C  X24:    JMP     X23
1797 0A2E 909C  E16A:   JMP     E16
1798
1799
1800                    ;*************************************
1801                    ;*     PUSH 1 ON ARITHMETIC STACK    *
1802                    ;*************************************
1803
1804 0A30 AAFD  LIT1:   ILD     LSTK(P2)
1805 0A32 AAFD          ILD     LSTK(P2)
1806 0A34 33            XPAL    P3
1807 0A35 C410          LDI     H(AESTK)
1808 0A37 37            XPAH    P3
1809 0A38 C400          LDI     0
1810 0A3A CBFF          ST      -1(P3)
1811 0A3C C401          LDI     1
1812 0A3E CBFE          ST      -2(P3)
1813 0A40 90EA          JMP     X24
1814
1815
1816                    ;*************************************
1817                    ;*      FOR-LOOP INITIALIZATION      *
1818                    ;*************************************
1819
1820                            .LOCAL
1821 0A42 C2FE  SAVFOR: LD      FORPTR(P2)      ;CHECK FOR FOR STACK
1822 0A44 E4A6          XRI     L(PCSTAK)       ; OVERFLOW
1823 0A46 9C04          JNZ     $1
1824 0A48 C40A          LDI     10
1825 0A4A 90E2  E17:    JMP     E16A
1826 0A4C E4A6  $1:     XRI     L(PCSTAK)
1827 0A4E 31            XPAL    P1              ;POINT P1 AT FOR STACK
1828 0A4F CAF1          ST      P1LOW(P2)       ; SAVING OLD P1
1829 0A51 C410          LDI     H(FORSTK)
1830 0A53 35            XPAH    P1
1831 0A54 CAF0          ST      P1HIGH(P2)
1832 0A56 C2FD          LD      LSTK(P2)        ;POINT P3 AT AE STACK
1833 0A58 33            XPAL    P3
1834 0A59 C410          LDI     H(AESTK)
1835 0A5B 37            XPAH    P3
1836 0A5C C3F9          LD      -7(P3)          ;GET VARIABLE INDEX
1837 0A5E CD01          ST      @1(P1)          ;SAVE ON FOR-STACK
1838 0A60 C3FC          LD      -4(P3)          ;GET L(LIMIT)
1839 0A62 CD01          ST      @1(P1)          ;SAVE
1840 0A64 C3FD          LD      -3(P3)          ;GET H(LIMIT)
1841 0A66 CD01          ST      @1(P1)          ;SAVE
1842 0A68 C3FE          LD      -2(P3)          ;GET L(STEP)
1843 0A6A CD01          ST      @1(P1)          ;SAVE
1844 0A6C C3FF          LD      -1(P3)          ;GET H(STEP)
1845 0A6E CD01          ST      @1(P1)          ;SAVE
1846 0A70 C2F1          LD      P1LOW(P2)       ;GET L(P1)
1847 0A72 CD01          ST      @1(P1)          ;SAVE
1848 0A74 C2F0          LD      P1HIGH(P2)      ;GET H(P1)
```

```
1849 0A76 CD01          ST      @1(P1)          ;SAVE
1850 0A78 35            XPAH    P1              ;RESTORE OLD P1
1851 0A79 C2F1          LD      P1LOW(P2)
1852 0A7B 31            XPAL    P1
1853 0A7C CAFE          ST      FORPTR(P2)      ;UPDATE FOR STACK PTR
1854 0A7E C7FC          LD      @-4(P3)
1855 0A80 33            XPAL    P3
1856 0A81 CAFD          ST      LSTK(P2)        ;UPDATE AE STACK PTR
1857 0A83 90A7  X25:    JMP     X24
1858
1859
1860                    ;*************************************
1861                    ;*     FIRST PART OF 'NEXT VAR'      *
1862                    ;*************************************
1863
1864                    .LOCAL
1865 0A85 C2FE  NEXTV:  LD      FORPTR(P2)      ;POINT P1 AT FOR STACK,
1866 0A87 E48A          XRI     L(FORSTK)       ; CHECKING FOR UNDERFLOW
1867 0A89 9C04          JNZ     $1
1868 0A8B C40B          LDI     11              ;REPORT ERROR
1869 0A8D 90BB          JMP     E17
1870 0A8F E48A  $1:     XRI     L(FORSTK)
1871 0A91 31            XPAL    P1
1872 0A92 CAF1          ST      P1LOW(P2)       ;SAVE OLD P1
1873 0A94 C410          LDI     H(FORSTK)
1874 0A96 35            XPAH    P1
1875 0A97 CAF0          ST      P1HIGH(P2)
1876 0A99 C2FD          LD      LSTK(P2)        ;POINT P3 AT AE STACK
1877 0A9B 33            XPAL    P3
1878 0A9C C410          LDI     H(AESTK)
1879 0A9E 37            XPAH    P3
1880 0A9F C7FF          LD      @-1(P3)         ;GET VARIABLE INDEX
1881 0AA1 E1F9          XOR     -7(P1)          ;COMPARE WITH INDEX
1882 0AA3 9804          JZ      $10             ; ON FOR STACK: ERROR
1883 0AA5 C40C          LDI     12              ; IF NOT EQUAL
1884 0AA7 90A1  E18:    JMP     E17
1885 0AA9 E1F9  $10:    XOR     -7(P1)          ;RESTORE INDEX
1886 0AAB 01            XAE                     ;SAVE IN EREG
1887 0AAC C280          LD      EREG(P2)        ;GET L(VARIABLE)
1888 0AAE 02            CCL
1889 0AAF F1FC          ADD     -4(P1)          ;ADD L(STEP)
1890 0AB1 CA80          ST      EREG(P2)        ;STORE IN VARIABLE
1891 0AB3 CB00          ST      (P3)            ; AND ON STACK
1892 0AB5 C601          LD      @1(P2)          ;INCREMENT RAM PTR
1893 0AB7 C280          LD      EREG(P2)        ;GET H(VARIABLE)
1894 0AB9 F1FD          ADD     -3(P1)          ;ADD H(STEP)
1895 0ABB CA80          ST      EREG(P2)        ;STORE IN VARIABLE
1896 0ABD CB01          ST      1(P3)           ; AND ON STACK
1897 0ABF C6FF          LD      @-1(P2)         ;RESTORE RAM POINTER
1898 0AC1 C1FA          LD      -6(P1)          ;GET L(LIMIT)
1899 0AC3 CB02          ST      2(P3)           ;PUT ON STACK
1900 0AC5 C1FB          LD      -5(P1)          ;GET H(LIMIT)
1901 0AC7 CB03          ST      3(P3)           ;PUT ON STACK
1902 0AC9 C1FD          LD      -3(P1)          ;GET H(STEP)
1903 0ACB 9410          JP      $2              ;IF NEGATIVE, INVERT
1904 0ACD C404          LDI     4               ; ITEMS ON A.E. STACK
1905 0ACF CAEB          ST      NUM(P2)         ;NUM = LOOP COUNTER
1906 0AD1 C701  $LOOP:  LD      @1(P3)          ;GET BYTE FROM STACK
1907 0AD3 E4FF          XRI     0FF             ;INVERT IT
1908 0AD5 CBFF          ST      -1(P3)          ;PUT BACK ON STACK
1909 0AD7 BAEB          DLD     NUM(P2)         ;DO UNTIL NUM = 0
1910 0AD9 9CF6          JNZ     $LOOP
1911 0ADB 9002          JMP     $3
1912 0ADD C704  $2:     LD      @4(P3)          ;UPDATE AE STACK POINTER
1913 0ADF 33    $3:     XPAL    P3
1914 0AE0 CAFD          ST      LSTK(P2)
1915 0AE2 C2F1          LD      P1LOW(P2)       ;RESTORE OLD P1
1916 0AE4 31            XPAL    P1
1917 0AE5 C2F0          LD      P1HIGH(P2)
1918 0AE7 35            XPAH    P1
1919 0AE8 9099  X26:    JMP     X25
1920
1921
1922                    ;*************************************
1923                    ;*    SECOND PART OF 'NEXT VAR'      *
1924                    ;*************************************
1925
1926 0AEA C2EF  NEXTV1: LD      LO(P2)          ;IS FOR-LOOP OVER WITH?
1927 0AEC 9808          JZ      $REDO           ;NO - REPEAT LOOP
1928 0AEE C2FE          LD      FORPTR(P2)      ;YES - POP FOR-STACK
1929 0AF0 02            CCL
1930 0AF1 F4F9          ADI     -7
1931 0AF3 CAFE          ST      FORPTR(P2)
1932 0AF5 3F            XPPC    P3              ;RETURN TO I.L. INTERPRETER
1933 0AF6 C2FE  $REDO:  LD      FORPTR(P2)      ;POINT P3 AT FOR STACK
1934 0AF8 33            XPAL    P3
1935 0AF9 C410          LDI     H(FORSTK)
1936 0AFB 37            XPAH    P3
1937 0AFC C3FF          LD      -1(P3)          ;GET OLD P1 OFF STACK
1938 0AFE 35            XPAH    P1
1939 0AFF C3FE          LD      -2(P3)
1940 0B01 31            XPAL    P1
1941 0B02 90E4          JMP     X26
1942 0B04 90A1  E19:    JMP     E18
1943
1944

1945                    ;*************************************
1946                    ;*     PRINT MEMORY AS STRING        *
1947                    ;*************************************
1948
1949                    ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
1950                    ;     'PRINT' '$' FACTOR
1951
1952                    .LOCAL
1953 0B06 C2EE  PSTRNG: LD      HI(P2)          ;POINT P1 AT STRING TO PRINT
1954 0B08 35            XPAH    P1
1955 0B09 C2EF          LD      LO(P2)
1956 0B0B 31            XPAL    P1
1957 0B0C               LDPI    P3,PUTC-1       ;POINT P3 AT PUTC ROUTINE
1958 0B12 C501  $1:     LD      @1(P1)          ;GET A CHARACTER
1959 0B14 E40D          XRI     0D              ;IS IT A CARRIAGE RETURN?
1960 0B16 98D0          JZ      X26             ;YES - WE'RE DONE
1961 0B18 E40D          XRI     0D              ;NO - PRINT THE CHARACTER
1962 0B1A 3F            XPPC    P3
1963 0B1B 06            CSA                     ;MAKE SURE NO ONE IS
1964 0B1C D420          ANI     020             ; TYPING ON THE TTY
1965 0B1E 9CF2          JNZ     $1              ; BEFORE REPEATING LOOP
1966 0B20 90C6          JMP     X26
1967
1968
1969                    ;*************************************
1970                    ;*          INPUT A STRING           *
1971                    ;*************************************
1972
1973                    ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
1974                    ;     'INPUT' '$' FACTOR
1975
1976 0B22 C2EE  ISTRNG: LD      HI(P2)          ;GET ADDRESS TO STORE THE
1977 0B24 37            XPAH    P3              ; STRING, PUT IT INTO P3
1978 0B25 C2EF          LD      LO(P2)
1979 0B27 33            XPAL    P3
1980 0B28 C501  $2:     LD      @1(P1)          ;GET A BYTE FROM LINE BUFFER
1981 0B2A CF01          ST      @1(P3)          ;PUT IT IN SPECIFIED LOCATION
1982 0B2C E40D          XRI     0D              ;DO UNTIL CHAR = CARR. RETURN
1983 0B2E 9CF8          JNZ     $2
1984 0B30 90B6  X27:    JMP     X26
1985
1986
1987                    ;*************************************
1988                    ;*   STRING CONSTANT ASSIGNMENT      *
1989                    ;*************************************
1990
1991                    ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
1992                    ;     '$' FACTOR '=' STRING
1993
1994                    .LOCAL
1995 0B32 C2EF  PUTSTR: LD      LO(P2)          ;GET ADDRESS TO STORE STRING,
1996 0B34 33            XPAL    P3              ; PUT IT INTO P3
1997 0B35 C2EE          LD      HI(P2)
1998 0B37 37            XPAH    P3
1999 0B38 C501  $LOOP:  LD      @1(P1)          ;GET A BYTE FROM STRING
2000 0B3A E422          XRI     '"'             ;CHECK FOR END OF STRING
2001 0B3C 980E          JZ      $END
2002 0B3E E42F          XRI     '"' ! 0D        ;MAKE SURE THERE'S NO CR
2003 0B40 9C04          JNZ     $1
2004 0B42 C407          LDI     7
2005 0B44 90BE          JMP     E19             ;ERROR IF CARRIAGE RETURN
2006 0B46 E40D  $1:     XRI     0D              ;RESTORE CHARACTER
2007 0B48 CF01          ST      @1(P3)          ;PUT IN SPECIFIED LOCATION
2008 0B4A 90EC          JMP     $LOOP           ;GET NEXT CHARACTER
2009 0B4C C40D  $END:   LDI     0D              ;APPEND CARRIAGE RETURN
2010 0B4E CB00          ST      (P3)            ; TO STRING
2011 0B50 90DE          JMP     X27
2012
2013
2014                    ;*************************************
2015                    ;*           MOVE STRING             *
2016                    ;*************************************
2017
2018                    ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
2019                    ;     '$' FACTOR '=' '$' FACTOR
2020
2021                    .LOCAL
2022 0B52 C2FD  MOVSTR: LD      LSTK(P2)        ;POINT P3 AT A.E. STACK
2023 0B54 33            XPAL    P3
2024 0B55 C410          LDI     H(AESTK)
2025 0B57 37            XPAH    P3
2026 0B58 C7FF          LD      @-1(P3)         ;GET ADDRESS OF SOURCE STRING
2027 0B5A 35            XPAH    P1              ; INTO P1
2028 0B5B C7FF          LD      @-1(P3)
2029 0B5D 31            XPAL    P1
2030 0B5E C7FF          LD      @-1(P3)         ;GET ADDRESS OF DESTINATION
2031 0B60 01            XAE                     ; STRING INTO P3
2032 0B61 C7FF          LD      @-1(P3)
2033 0B63 33            XPAL    P3
2034 0B64 CAFD          ST      LSTK(P2)        ;UPDATE STACK POINTER
2035 0B66 40            LDE
2036 0B67 37            XPAH    P3
2037 0B68 C501  $LOOP:  LD      @1(P1)          ;GET A SOURCE CHARACTER
2038 0B6A CF01          ST      @1(P3)          ;SEND IT TO DESTINATION
2039 0B6C E40D          XRI     0D              ;REPEAT UNTIL CARRIAGE RET.
2040 0B6E 98C0          JZ      X27
2041 0B70 06            CSA                     ; OR KEYBOARD INTERRUPT
2042 0B71 D420          ANI     020
2043 0B73 9CF3          JNZ     $LOOP
2044 0B75 90B9          JMP     X27
2045
2046
2047                    ;*************************************
2048                    ;*     PUT PAGE NUMBER ON STACK      *
2049                    ;*************************************
2050
2051 0B77 AAFD  PUTPGE: ILD     LSTK(P2)
2052 0B79 AAFD          ILD     LSTK(P2)
2053 0B7B 33            XPAL    P3
2054 0B7C C410          LDI     H(AESTK)
2055 0B7E 37            XPAH    P3
2056 0B7F C2F6          LD      PAGE(P2)
2057 0B81 CBFE          ST      -2(P3)
2058 0B83 C400          LDI     0
2059 0B85 CBFF          ST      -1(P3)
2060 0B87 90A7          JMP     X27
2061
2062
2063                    ;*************************************
2064                    ;*         ASSIGN NEW PAGE           *
2065                    ;*************************************
2066
2067                    .LOCAL
2068 0B89 C2EF  NUPAGE: LD      LO(P2)          ;GET PAGE # FROM STACK,
2069 0B8B D407          ANI     7               ; GET THE LOW 3 BITS
2070 0B8D 9C02          JNZ     $0              ;PAGE 0 BECOMES PAGE 1
2071 0B8F C401          LDI     1
2072 0B91 CAF6  $0:     ST      PAGE(P2)
2073 0B93 3F            XPPC    P3              ;RETURN
2074
2075
2076                    ;*************************************
2077                    ;*         FIND START OF PAGE        *
2078                    ;*************************************
2079
2080                    ; THIS ROUTINE COMPUTES THE START OF THE CURRENT TEXT PAGE,
2081                    ; STORING THE ADDRESS IN TEMP2(P2) [THE HIGH BYTE], AND
2082                    ; TEMP3(P2) [THE LOW BYTE].
2083
2084 0B94 C2F6  FNDPGE: LD      PAGE(P2)
2085 0B96 E401          XRI     1               ;SPECIAL CASE IS PAGE 1, BUT
2086 0B98 9C09          JNZ     $1              ; OTHERS ARE CONVENTIONAL
2087 0B9A C411          LDI     H(PGM)          ;PAGE 1 STARTS AT 'PGM'
2088 0B9C CAE9          ST      TEMP2(P2)
2089 0B9E C420          LDI     L(PGM)
2090 0BA0 CAE8          ST      TEMP3(P2)
2091 0BA2 3F            XPPC    P3              ;RETURN
2092 0BA3 E401  $1:     XRI     1               ;RESTORE PAGE #
2093 0BA5 01            XAE                     ;SAVE IT
2094 0BA6 C404          LDI     4               ;LOOP COUNTER = 4
2095 0BA8 CAEB          ST      NUM(P2)
2096 0BAA 40    $LOOP:  LDE                     ;MULTIPLY PAGE# BY 16
2097 0BAB 02            CCL
2098 0BAC 70            ADE
2099 0BAD 01            XAE
2100 0BAE BAEB          DLD     NUM(P2)
2101 0BB0 9CF8          JNZ     $LOOP
2102 0BB2 40            LDE
2103 0BB3 CAE9          ST      TEMP2(P2)       ;TEMP2 HAS HIGH BYTE
2104 0BB5 C402          LDI     2               ; OF ADDRESS NOW
2105 0BB7 CAE8          ST      TEMP3(P2)       ;LOW BYTE IS ALWAYS 2
2106 0BB9 3F            XPPC    P3
2107
2108
2109                    ;*************************************
2110                    ;*    MOVE CURSOR TO NEW PAGE        *
2111                    ;*************************************
2112
2113 0BBA C2E9  CHPAGE: LD      TEMP2(P2)       ;PUT START OF PAGE
2114 0BBC 35            XPAH    P1              ; INTO P1.  THIS ROUTINE
```

S.D. SALES CO.

P.O. BOX 28810 -E
DALLAS, TEXAS 75228

Z–80 CPU CARD KIT FOR IMSAI/ALTAIR

$149. KIT

From the same people who brought you the $89.95 4K RAM kit. We were not the first to introduce an IMSAI/ALTAIR compatible Z–80 card, but we do feel that ours has the best design and quality at the lowest price.

The advanced features of the Z–80 such as an expanded set of 158 instructions, 8080A software compatibility, and operation from a single 5VDC supply, are all well known. What makes our card different is the extra care we took in the hardware design. The CPU card will always stop on an M1 state. We also generate TRUE SYNC on card, to insure that the rest of your system functions properly. Dynamic memory refresh and NMI are brought out for your use. Believe it or not, not all of our competitors have gone to the extra trouble of doing this.

As always, this kit includes all parts, all sockets, and complete instructions for ease of assembly. Because of our past experience with our 4K kit we suggest that you order early. All orders will be shipped on a strict first come basis. Dealers inquiries welcome on this item.

Kit shipped with 2 MHZ crystals for existing 500NS memory. Easily modified for faster RAM chips when the prices come down.

Z–80 Manual – $7.50 Separately.

Kit includes Zilog Manual and all parts.

JUMBO LED CAR CLOCK

$16.95 KIT

You requested it! Our first DC operated clock kit. Professionally engineered from scratch to be a DC operated clock. Not a makeshift kluge as sold by others. Features: Bowmar 4 digit .5 inch LED array, Mostek 50252 super clock chip, on board precision time base, 12 or 24 hour real time format, perfect for cars, boats, vans, etc. Kit contains PC Board and all other parts needed (except case). 50,000 satisfied clock kit customers cannot be wrong!

FOR ALARM OPTION ADD $1.50
FOR XFMR FOR AC OPERATION ADD $1.50

60 HZ CRYSTAL TIME BASE FOR DIGITAL CLOCKS
S.D. SALES EXCLUSIVE!

KIT FEATURES:

A. 60HZ output with accuracy comparable to a digital watch.
B. Directly interfaces with all MOS Clock Chips.
C. Super low power consumption. (1.5 ma typ.) **$5.95 or**
D. Uses latest MOS 17 stage divider IC. **2/$10.**
E. Eliminates forever the problem of AC line glitches.
F. Perfect for cars, boats, campers, or even for portable clocks at ham field days.
G. Small Size, can be used in existing enclosures.

KIT INCLUDES CRYSTAL, DIVIDER IC, PC BOARD PLUS ALL OTHER NECESSARY PARTS & SPECS

50HZ CRYSTAL TIME BASE KIT – $6.95

All the features of our 60HZ kit but has 50HZ output. For use with clock chips like the 50252 that require 50HZ to give 24 hour time format.

THIS MONTH'S SPECIALS!

SPECIAL

300.00 KHZ CRYSTAL – $1.50
8080A – CPU CHIP by AMD – $19.95
82S129 – 256 x 4 PROM – $2.50
N.S. 8865 OCTAL DARLINGTON DRIVERS
3 for $1.00
Z-80 – CPU by ZILOG – $69.95
MM5204 – 4K EPROM – $7.95

SPECIAL

Prices in effect this month ONLY!

4K LOW POWER RAM BOARD KIT
THE WHOLE WORKS – $89.95

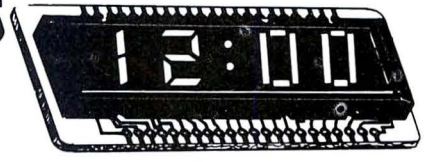
Imsai and Altair 8080 plug in compatible. Uses low power static 21L02–1 500ns. RAM's, which are included. Fully buffered, drastically reduced power consumption, on board regulated, all sockets and parts included. Premium quality plated thru PC Board.

7400–19c	7411–29c	7451–19c	7490–65c	74153–75c
74LS00–49c	7413–50c	7453–19c	74LS90–95c	74154-1.00
7402–19c	7416–69c	7473–39c	7492–75c	74157–75c
74LS02–49c	7420–19c	7474–35c	7493–69c	74161–95c
7404–19c	7430–19c	74LS74-59c	7495–75c	74164-1.10
74L04–29c	7432–34c	7475–69c	7496–89c	74165-1.10
74S04–44c	7437–39c	7476–35c	74121-38c	74174–95c
74LS04-49c	7438–39c	7480–49c	74123-65c	74181-2.50
7406–29c	7440–19c	7483–95c	74132-1.70	74191-1.25
7408–19c	7447–85c	7485–95c	74S138-1.95	74192-1.25
7410–19c	7448–85c	7486–45c	74141-75c	74193-1.00
TTL INTEGRATED CIRCUITS				74195–69c

STICK IT! in your clock in your DVM, etc.!

Huge Special Purchase Not Factory Seconds As sold by others!

$3.95



4 JUMBO .50" DIGITS ON ONE STICK! (with colons and AM/PM Indicator)

BUY 3 for $10.

BOWMAR 4 DIGIT LED READOUT ARRAY

The Bowmar Opto-Stick. The best readout bargain we have ever offered. Has four common cathode jumbo digits with all segments and cathodes brought out. Increased versatility since any of the digits may be used independently to fit your applications. Perfect for any clock chip, especially direct drive units like 50380 or 7010. Also use in freq. counters, DVM's, etc. For 12 or 24 hour format.

UP YOUR COMPUTER!
21L02–1 1K LOW POWER 500 NS STATIC RAM
TIME IS OF THE ESSENCE!

And so is power. Not only are our RAM'S faster than a speeding bullet but they are now very low power. We are pleased to offer prime new 21L02–1 low power and super fast RAM's. Allows you to STRETCH your power supply farther and at the same time keep the wait light off. **8 for $12.95**

$12.95 S.D. SALES EXCLUSIVE! MOS 6 DIGIT UP-DOWN COUNTER **$12.95**

40 PIN DIP. Everything you ever wanted in a counter chip. Features: Direct LED segment drive, single power supply (12 VDC TYP.), six decades up/down, pre-loadable counter, separate pre-loadable compare register with compare output, BCD AND seven segment outputs, internal scan oscillator, CMOS compatible, leading zero blanking. 1MHZ. count input frequency. Very limited qty. WITH DATA SHEET

WESTERN DIGITAL UART

No. TR1602B. 40 pin DIP
This is a very powerful and popular part.
NEW–$6.95 with data
LIMITED QUANTITY

SLIDE SWITCH ASSORTMENT

Our best seller. Includes miniature and standard sizes, single and multi-position units. All new, first quality, name brand. Try one package and you'll reorder more. **SPECIAL 12/$1.**

RESISTOR ASSORTMENT

¼ W 5% and 10% PC leads. A good mix of values. 200/$2.

MOTOROLA POWER DARLINGTON

Back in Stock!

Like MJ3001. NPN 80V. 10A. HFE 6000 TYP. TO–3 case. We include a free 723C volt reg. with schematic for power supply. SPECIAL–$1.99

FAIRCHILD BIG LED READOUTS

A big .50 inch easy to read character. Now available in either common anode or common cathode. Take your pick. Super low current drain, only 5MA per segment typical.

FND 510 Common Anode
FND 503 Common Cathode
PRICE SLASHED! 59c each

CALL YOUR BANK AMERICARD OR MASTER CHARGE ORDER IN ON OUR CONTINENTAL UNITED STATES TOLL FREE WATTS:

1–800–527-3460

Texas Residents Call Collect

214/271-0022

TERMS:

Money Back Guarantee. No COD. Texas Residents add 5% tax. Add 5% of order for postage and handling. Orders under $10. add 75c. Foreign orders: US Funds ONLY!

S.D. SALES CO.
P.O. BOX 28810 · E
Dallas, Texas 75228

For orders over $15.00 Choose $1.00 FREE mdse.

```
0BBD C2E8          LD      TEMP3(P2)       ; MUST BE CALLED RIGHT
0BBF 31            XPAL    P1              ; AFTER 'FNDPGE'
0BC0 3F            XPPC    P3              ;RETURN


                   ;************************************
                   ;*       DETERMINE CURRENT PAGE     *
                   ;************************************

0BC1 35    DETPGE: XPAH    P1              ;CURRENT PAGE IS HIGH
0BC2 01            XAE                     ; PART OF CURSOR DIVIDED
0BC3 40            LDE                     ; BY 16
0BC4 35            XPAH    P1
0BC5 40            LDE
0BC6 1C            SR
0BC7 1C            SR
0BC8 1C            SR
0BC9 1C            SR
0BCA CAF6          ST      PAGE(P2)
0BCC 3F            XPPC    P3              ;RETURN


                   ;************************************
                   ;*       CLEAR CURRENT PAGE         *
                   ;************************************

0BCD C2E9  NEWPGM: LD      TEMP2(P2)       ;POINT P1 AT CURRENT PAGE
0BCF 35            XPAH    P1
0BD0 C2E8          LD      TEMP3(P2)
0BD2 31            XPAL    P1
0BD3 C40D          LDI     0D              ;PUT DUMMY END-OF-LINE
0BD5 C9FF          ST      -1(P1)          ; JUST BEFORE TEXT
0BD7 C4FF          LDI     -1              ;PUT -1 AT START OF TEXT
0BD9 C900          ST      (P1)
0BDB C901          ST      1(P1)
0BDD 3F            XPPC    P3              ;RETURN


                   ;************************************
                   ;*      FIND LINE NUMBER IN TEXT    *
                   ;************************************

                   ;INPUTS:  THE START OF THE CURRENT PAGE IN TEMP2 AND TEMP3,
                   ;         THE LINE NUMBER TO LOOK FOR IN LO AND HI.
                   ;OUPUTS:  THE ADDRESS OF THE FIRST LINE IN THE NIBL TEXT
                   ;         WHOSE LINE NUMBER IS GREATER THAN OR EQUAL TO THE
                   ;         NUMBER IN HI AND LO, RETURNED IN ADRLO AND ADRHI

                   .LOCAL
0BDE C2E9  FNDLBL: LD      TEMP2(P2)       ;POINT P1 AT START OF TEXT
0BE0 35            XPAH    P1
0BE1 C2E8          LD      TEMP3(P2)
0BE3 31            XPAL    P1
0BE4 C100  $1:     LD      (P1)            ;HAVE WE HIT END OF TEXT?
0BE6 E4FF          XRI     0FF
0BE8 9412          JP      $2              ;YES - STOP LOOKING
0BEA 03            SCL                     ;NO - COMPARE LINE NUMBERS
0BEB C101          LD      1(P1)           ; BY SUBTRACTING
0BED FAEF          CAD     LO(P2)
0BEF C100          LD      0(P1)
0BF1 FAEE          CAD     HI(P2)          ;IS TEXT LINE # >= LINE #?
0BF3 9407          JP      $2              ;YES - STOP LOOKING
0BF5 C102          LD      2(P1)           ;NO - TRY NEXT LINE IN TEXT
0BF7 01            XAE
0BF8 C580          LD      @EREG(P1)       ; SKIP LENGTH OF LINE
0BFA 90E8          JMP     $1
0BFC 31    $2:     XPAL    P1              ;SAVE ADDRESS OF FOUND LINE
0BFD CAF3          ST      LABLLO(P2)      ; IN LABLHI AND LABLLO
0BFF 31            XPAL    P1
0C00 35            XPAH    P1
0C01 CAF2          ST      LABLHI(P2)
0C03 35            XPAH    P1
0C04 C2EF          LD      LO(P2)          ;WAS THERE AN EXACT MATCH?
0C06 E101          XOR     1(P1)
0C08 9C07          JNZ     $3
0C0A C2EE          LD      HI(P2)
0C0C E100          XOR     0(P1)
0C0E 9C01          JNZ     $3              ;NO - FLAG THE ADDRESS
0C10 3F            XPPC    P3              ;YES - RETURN NORMALLY
0C11 C2F2  $3      LD      LABLHI(P2)      ;SET SIGN BIT OF HIGH PART
0C13 DC80          ORI     080             ; OF ADDRESS TO INDICATE
0C15 CAF2          ST      LABLHI(P2)      ; INEXACT MATCH OF LINE #'S
0C17 3F            XPPC    P3


                   .PAGE   ' I. L. MACROS'

                   ;************************************
                   ;*          I. L. MACROS            *
                   ;************************************

                   .LOCAL

      2000 $TSTBIT =       TSTBIT*256
      8000 $CALBIT =       CALBIT*256
      4000 $JMPBIT =       JMPBIT*256

                   .MACRO  TST,FAIL,A,B
                   .DBYTE  $TSTBIT!FAIL
                   .IF     #=2
                   .BYTE   'A'!080
                   .ELSE
                   .ASCII  'A'
                   .BYTE   'B'!080
                   .ENDIF
                   .ENDM

                   .MACRO  TSTCR,FAIL
                   .DBYTE  $TSTBIT!FAIL
                   .BYTE   0D!080
                   .ENDM

                   .MACRO  TSTV,FAIL
                   .ADDR   TSTVAR
                   .DBYTE  FAIL
                   .ENDM

                   .MACRO  TSTN,FAIL
                   .ADDR   TSTNUM
                   .DBYTE  FAIL
                   .ENDM

                   .MACRO  JUMP,ADR
                   .DBYTE  $JMPBIT!ADR
                   .ENDM

                   .MACRO  CALL,ADR
                   .DBYTE  $CALBIT!ADR
                   .ENDM

                   .MACRO  DO
                   .MLOC   I
                   .SET    I,1
                   .DO     #
                   .ADDR   #I
                   .SET    I,I+1
                   .ENDDO
                   .ENDM
                   .PAGE   ' I. L. TABLE'

                   ;************************************
                   ;*          I. L. TABLE             *
                   ;************************************

0C18 START:  DO      NLINE
0C1A PROMPT: DO      GETL
0C1C         TSTCR   PRMPT1
0C1F         JUMP    PROMPT
0C21 PRMPT1: TSTN    LIST
0C25         DO      FNDPGE,XCHGP1,POPAE,FNDLBL,INSRT
0C2F         JUMP    PROMPT

0C31 LIST:   TST     RUN,'LIS','T'
0C37         DO      FNDPGE
0C39         TSTN    LIST1
0C3D         DO      POPAE,FNDLBL
0C41         JUMP    LIST2
0C43 LIST1:  DO      CHPAGE
0C45 LIST2:  DO      LST
0C47 LIST3:  CALL    PRNUM
0C49         DO      LST3
0C4B         JUMP    START

0C4D RUN:    TST     CLR,'RU','N'
0C52         DO      DONE
0C54 BEGIN:  DO      FNDPGE,CHPAGE,STRT,NXT

0C5C CLR:    TST     NEW,'CLEA','R'
0C63         DO      DONE,CLEAR,NXT

0C69 NEW:    TST     STMT,'NE','W'
0C6E         TSTN    DFAULT
0C72         JUMP    NEW1
0C74 DFAULT: DO      LIT1
0C76 NEW1:   DO      DONE,POPAE,NUPAGE,FNDPGE,NEWPGM,NXT

0C82 STMT:   TST     LET,'LE','T'
0C87 LET:    TSTV    AT
0C8B         TST     SYNTAX,'='
0C8E         CALL    RELEXP
0C90         DO      STORE,DONE,NXT
0C96 AT:     TST     IF,'@'
0C99         CALL    FACTOR
0C9B         TST     SYNTAX,'='
0C9E         CALL    RELEXP
0CA0         DO      MOVE,DONE,NXT

0CA6 IF:     TST     UNT,'I','F'
0CAA         CALL    RELEXP
0CAC         TST     IF1,'THE','N'
0CB2 IF1:    DO      POPAE,CMPR
0CB6         JUMP    STMT

0CB8 UNT:    TST     DO,'UNTI','L'
0CBF         DO      CKMODE
0CC1         CALL    RELEXP
0CC3         DO      DONE,POPAE,UNTIL,DETPGE,NXT

0CCD DO:     TST     GOTO,'D','O'
0CD1         DO      CKMODE,DONE,SAVEDO,NXT

0CD9 GOTO:   TST     RETURN,'G','O'
0CDD         TST     GOSUB,'T','O'
0CE1         CALL    RELEXP
0CE3         DO      DONE
0CE5         JUMP    GO1
0CE7 GOSUB:  TST     SYNTAX,'SU','B'
0CEC         CALL    RELEXP
0CEE         DO      DONE,SAV
0CF2 GO1:    DO      FNDPGE,POPAE,FNDLBL,XFER,NXT

0CFC RETURN: TST     NEXT,'RETUR','N'
0D04         DO      DONE,RSTR,DETPGE,NXT

0D0C NEXT:   TST     FOR,'NEX','T'
0D12         DO      CKMODE
0D14         TSTV    SYNTAX
0D18         DO      DONE,NEXTV
0D1C         CALL    GTROP
0D1E         DO      POPAE,NEXTV1,DETPGE,NXT

0D26 FOR:    TST     STAT,'FO','R'
0D2B         DO      CKMODE
0D2D         TSTV    SYNTAX
0D31         TST     SYNTAX,'='
0D34         CALL    RELEXP
0D36         TST     SYNTAX,'T','O'
0D3A         CALL    RELEXP
0D3C         TST     FOR1,'STE','P'
0D42         CALL    RELEXP
0D44         JUMP    FOR2
0D46 FOR1:   DO      LIT1
0D48 FOR2:   DO      DONE,SAVFOR,STORE,NXT

0D50 STAT:   TST     PGE,'STA','T'
0D56         TST     SYNTAX,'='
0D59         CALL    RELEXP
0D5B         DO      POPAE,MOVESR
0D5F         DO      DONE,NXT

0D63 PGE:    TST     DOLLAR,'PAG','E'
0D69         TST     SYNTAX,'='
0D6C         CALL    RELEXP
0D6E         DO      DONE,POPAE,NUPAGE,FNDPGE,CHPAGE,NXT

0D7A DOLLAR: TST     PRINT,'$'
0D7D         CALL    FACTOR
0D7F         TST     SYNTAX,'='
0D82         TST     DOLR1,'"'
0D85         DO      POPAE,PUTSTR
0D89         JUMP    DOLR2
0D8B DOLR1:  TST     SYNTAX,'$'
0D8E         CALL    FACTOR
0D90         DO      XCHGP1,MOVSTR,XCHGP1
0D96 DOLR2:  DO      DONE,NXT

0D9A PRINT:  TST     INPUT,'P','R'
0D9E         TST     PR1,'IN','T'
0DA3 PR1:    TST     PR2,'"'
0DA6         DO      PRS
0DA8         JUMP    COMMA
0DAA PR2:    TST     PR3,'$'
0DAD         CALL    FACTOR
0DAF         DO      XCHGP1,POPAE,PSTRNG,XCHGP1
0DB7         JUMP    COMMA
0DB9 PR3:    CALL    RELEXP
0DBB         CALL    PRNUM
0DBD COMMA:  TST     PR4,','
0DC0         JUMP    PR1
```

DIODES/ZENERS

1N914	100v	10mA	.05
1N4004	400v	1A	.08
1N4005	600v	1A	.08
1N4007	1000v	1A	.15
1N4148	75v	10mA	.03
1N753A	6.2v	z	.25
1N758A	10v	z	.25
1N759A	12v	z	.25
1N4733	5.1v	z	.25
1N5243	13v	z	.25
1N5244B	14v	z	.25
1N5245B	15v	z	.25

SOCKETS/BRIDGES

8-pin	pcb	.25	ww	.45
14-pin	pcb	.25	ww	.40
16-pin	pcb	.25	ww	.40
18-pin	pcb	.25	ww	.75
22-pin	pcb	.45	ww	.75
24-pin	pcb	.35	ww	1.25
28-pin	pcb	.35	ww	1.45
40-pin	pcb	.50	ww	1.95

Molex pins .01 To-3 Sockets .25

2 Amp Bridge 100-prv 1.20

25 Amp Bridge 200-prv 2.50

TRANSISTORS, LEDS, etc.

2N2222	NPN			.10
2N2907	PNP			.15
2N3740	PNP	1A	60v	.25
2N3906	PNP			.10
2N3055	NPN	15A	60v	.50

LED Green, Red, Clear .15

D.L. 747 7 seg 5/8" high 1.95

XAN72 7 seg com-anode 1.50

FND 359 Red 7 seg com-cathode 1.00

C MOS

4000	.20
4001	.20
4002	.25
4004	4.95
4006	1.20
4007	.40
4008	1.20
4009	.25
4010	.45
4011	.20
4012	.25
4013	.40
4014	1.10
4015	.95
4016	.35
4017	1.10
4018	1.10
4019	.70
4020	.85
4021	1.35
4022	1.15
4023	.25
4024	.95
4025	.35
4026	1.95
4027	.50
4028	.95
4030	.45
4033	1.95
4034	2.45
4035	1.25
4040	1.35
4042	.95
4043	1.25
4044	.95
4046	1.50
4049	.80
4050	.70
4066	1.35
4069	.40
4071	.35
4082	.45

— T T L —

7400	.15
7401	.15
7402	.20
7403	.25
7404	.15
7405	.25
7406	.45
7407	.55
7408	.25
7409	.15
7410	.15
7411	.25
7412	.30
7413	.65
7414	1.10
7416	.25
7417	.50
7420	.15
7426	.40
7427	.45
7430	.15
7432	.45
7437	.45
7438	.35
7440	.25
7441	1.15
7442	.65
7443	.95
7444	.95
7445	.95
7446	.95
7447	.95
7448	1.20
7450	.25
7451	.25
7453	.25
7454	.25
7460	.40
7470	.45
7472	.45
7473	.35

7474	.40
7475	.45
7476	.20
7480	.65
7483	1.00
7485	1.05
7486	.40
7489	2.50
7490	.40
7491	1.15
7492	.95
7493	.45
7494	1.25
7495	.85
7496	.95
74100	1.85
74107	.45
74121	.40
74122	.55
74123	.55
74125	.45
74132	1.35
74141	1.30
74150	1.00
74151	.95
74153	.95
74154	.75
74156	1.15
74157	.75
74161	1.25
74163	1.25
74164	.95
74165	1.50
74166	1.35
74175	.95
74176	1.25
74180	.85
74181	3.25
74182	.95
74190	1.75
74192	1.65

74193	.85
74194	1.45
74195	.95
74196	1.50
74197	1.25
74198	2.35
74367	.85
75108A	.35
75110	.35
75491	.50
75492	.50
74H00	.25
74H01	.25
74H04	.25
74H05	.25
74H15	.30
74H20	.30
74H22	.40
74H30	.25
74H40	.25
74H52	.15
74H53J	.25
74H55	.25
74H72	.55
74H101	.75
74H103	.75
74H106	.95
74L00	.35
74L02	.35
74L03	.30
74L10	.35
74L30	.45
74L47	1.95
74L75	.55

74S00	.55
74S02	.55
74S03	.50
74S10	.45
74S11	.45
74S20	.50
74S40	.30
74S51	.45
74S64	.30
74S74	.50
74S112	1.50
74S133	.45
74S140	.75
74S151A	.45
74S153	.45
74S158	.45
74S194	1.50
74S257 (8123)	.25
74LS00	.45
74LS01	.45
74LS02	.45
74LS04	.55
74LS08	.45
74LS09	.45
74LS10	.45
74LS11	.45
74LS20	.50
74LS21	.25
74LS22	.25
74LS32	.55
74LS37	.40
74LS40	.55
74LS42	1.75
74LS52	1.45
74LS74	.95
74LS90	1.30
74LS93	1.00
74LS107	.95
74LS153	1.20
74LS157	.85
74LS164	1.90
74LS368	.70

9000 SERIES

9301	1.00
9309	.45
9602	1.50

MEMORY, CLOCKS

74S188 (8223)	3.00
8080	26.50
MM1702A	10.50
MM5314	3.50
MM5316	3.95
2102-1	1.75
2102-L1	1.95
TR 1602A	6.95

LINEARS, REGULATORS, etc.

MCT2	.95
8038	3.95
LM201AH	.75
LM301AH	.25
LM308AH	1.00
LM309H	.65
LM309K	.90
LM310	1.15
LM311	1.35

LM320K5	1.65
LM320K12	1.65
LM320T12	1.65
LM320T15	1.65
LM339	1.65
7805(340T-5)	1.00
LM340T-12	1.25
LM340T-15	1.25
LM340T-18	1.65

LM340T-24	1.25
LM340K-12	2.15
LM340K-15	1.65
LM340K-18	1.65
LM340K-24	1.25
LM373	1.95
LM380	.95
LM709	.30
LM711	.45

LM723	.45
LM725	1.95
LM739	1.50
LM741 8-14	.25
LM747	1.10
LM1307	1.25
LM1458	.95
LM3900	.65
LM75451	.65
NE555	.50
NE556	1.10
NE565	.95
NE566	1.75
NE567	1.35
SN72720	.35
SN72820	.35

```
0ODC2          PR4:    TST     PR5,'/'
0ODC5                  JUMP    PR6
0ODC7          PR5:    DO      NLINE
0ODC9          PR6:    DO      DONE,NXT

0ODCD          INPUT:  TST     END,'INPU','T'
0ODD4                  DO      CKMODE
0ODD6                  TSTV    IN2
0ODDA                  DO      XCHGP1,GETL
0ODDE          IN1:    CALL    RELEXP
0ODE0                  DO      STORE,XCHGP1
0ODE4                  TST     IN3,','
0ODE7                  TSTV    SYNTAX
0ODEB                  DO      XCHGP1
0ODED                  TST     SYNTAX,','
0ODF0                  JUMP    IN1
0ODF2          IN2:    TST     SYNTAX,'$'
0ODF5                  CALL    FACTOR
0ODF7                  DO      XCHGP1,GETL,POPAE,ISTRNG,XCHGP1
0E01           IN3:    DO      DONE,NXT

0E05           END:    TST     ML,'EN','D'
0E0A                   DO      DONE,BREAK

0E0E           ML:     TST     REM,'LIN','K'
0E14                   CALL    RELEXP
0E16                   DO      DONE,XCHGP1,POPAE,CALLML,XCHGP1,NXT

0E22           REM:    TST     SYNTAX,'RE','M'
0E27                   DO      IGNORE,NXT

0E2B           SYNTAX: DO      ERR
0E2D           ERRNUM: CALL    PRNUM
0E2F                   DO      FIN

               ; NOTE: EACH RELATIONAL OPERATOR (EQ, LEQ, ETC.) DOES AN
               ; AUTOMATIC 'RTN' (THIS SAVES VALUABE BYTES!)

0E31           RELEXP: CALL    EXPR
0E33                   TST     REL1,'='
0E36                   CALL    EXPR
0E38                   DO      EQ
0E3A           REL1:   TST     REL4,'<'
0E3D                   TST     REL2,'='
0E40                   CALL    EXPR
0E42                   DO      LEQ
0E44           REL2:   TST     REL3,'>'
0E47                   CALL    EXPR
0E49                   DO      NEQ
0E4B           REL3:   CALL    EXPR
0E4D                   DO      LSS
0E4F           REL4:   TST     RETEXP,'>'
0E52                   TST     REL5,'='
0E55                   CALL    EXPR
0E57                   DO      GEQ
0E59           REL5:   CALL    EXPR
0E5B           GTROP:  DO      GTR

0E5D           EXPR:   TST     EX1,'-'
0E60                   CALL    TERM
0E62                   DO      NEG
0E64                   JUMP    EX3
0E66           EX1:    TST     EX2,'+'
0E69           EX2:    CALL    TERM
0E6B           EX3:    TST     EX4,'+'
0E6E                   CALL    TERM
0E70                   DO      ADD
0E72                   JUMP    EX3
0E74           EX4:    TST     EX5,'-'
0E77                   CALL    TERM
0E79                   DO      SUB
0E7B                   JUMP    EX3
0E7D           EX5:    TST     RETEXP,'O','R'
0E81                   CALL    TERM
0E83                   DO      OROP
0E85                   JUMP    EX3
0E87           RETEXP: DO      RTN

0E89           TERM:   CALL    FACTOR
0E8B           T1:     TST     T2,'*'
0E8E                   CALL    FACTOR
0E90                   DO      MUL
0E92                   JUMP    T1
0E94           T2:     TST     T3,'/'
0E97                   CALL    FACTOR
0E99                   DO      DIV
0E9B                   JUMP    T1
0E9D           T3:     TST     RETEXP,'AN','D'
0EA2                   CALL    FACTOR
0EA4                   DO      ANDOP
0EA6                   JUMP    T1

0EA8           FACTOR: TSTV    F1
0EAC                   DO      IND,RTN
0EB0           F1:     TSTN    F2
0EB4                   DO      RTN
0EB6           F2:     TST     F3,'#'
0EB9                   DO      HEX,RTN
0EBD           F3:     TST     F4,'('
0EC0                   CALL    RELEXP
0EC2                   TST     SYNTAX,')'
0EC5                   DO      RTN
0EC7           F4:     TST     F5,'@'
0ECA                   CALL    FACTOR
0ECC                   DO      EVAL,RTN
0ED0           F5:     TST     F6,'NO','T'
0ED5                   CALL    FACTOR
0ED7                   DO      NOTOP,RTN
0EDB           F6:     TST     F7,'STA','T'
0EE1                   DO      STATUS,RTN
0EE5           F7:     TST     F8,'TO','P'
0EEA                   DO      FNDPGE,TOP,RTN
0EF0           F8:     TST     F9,'MO','D'
0EF5                   CALL    DOUBLE
0EF7                   DO      DIV,MODULO,RTN
0EFD           F9:     TST     F10,'RN','D'
0F02                   CALL    DOUBLE
0F04                   DO      RANDOM,SUB,ADD,DIV,MODULO,ADD,RTN
0F12           F10:    TST     SYNTAX,'PAG','E'
0F18                   DO      PUTPGE,RTN

0F1C           DOUBLE: TST     SYNTAX,'('
0F1F                   CALL    RELEXP
0F21                   TST     SYNTAX,','
0F24                   CALL    RELEXP
0F26                   TST     SYNTAX,')'
0F29                   DO      RTN

0F2B           PRNUM:  DO      XCHGP1,PRN
0F2F           PRNUM1: DO      DIV,PRN1,XCHGP1,RTN
                       PAGE    'ERROR MESSAGES'

               ;*************************************
               ;*            ERROR MESSAGES         *
               ;*************************************

               ;*************************************

                       .MACRO  MESSAGE,A,B
                       .ASCII  'A'
                       .BYTE   'B'!080
                       .ENDM

0F37           MESGS:  MESSAGE ' ERRO','R'     ; 1
0F3D                   MESSAGE 'ARE','A'       ; 2
0F41                   MESSAGE 'STM','T'       ; 3
0F45                   MESSAGE 'CHA','R'       ; 4
0F49                   MESSAGE 'SNT','X'       ; 5
0F4D                   MESSAGE 'VAL','U'       ; 6
0F51                   MESSAGE 'END','"'       ; 7
0F55                   MESSAGE 'NOG','O'       ; 8
0F59                   MESSAGE 'RTR','N'       ; 9
0F5D                   MESSAGE 'NES','T'       ; 10
0F61                   MESSAGE 'NEX','T'       ; 11
0F65                   MESSAGE 'FO','R'        ; 12
0F68                   MESSAGE 'DIV','O'       ; 13
0F6C                   MESSAGE 'BR','K'        ; 14
0F6F                   MESSAGE 'UNT','L'       ; 15
                        PAGE   ' TELETYPE ROUTINES'

               ;*************************************
               ;*     GET CHARACTER AND ECHO IT     *
               ;*************************************
               ;
                       .LOCAL
0F73 C408      GECO:   LDI     8               ;SET COUNT = 8
0F75 CAEB              ST      NUM(P2)
0F77 06                CSA                     ;SET READER RELAY
0F78 DC02              ORI     2
0F7A 07                CAS
0F7B 06        $1:     CSA                     ;WAIT FOR START BIT
0F7C D420              ANI     020
0F7E 9CFB              JNZ     $1              ;NOT FOUND
0F80 C457              LDI     87              ;DELAY 1/2 BIT TIME
0F82 8F04              DLY     4
0F84 06                CSA                     ;IS START BIT STILL THERE?
0F85 D420              ANI     020
0F87 9CF2              JNZ     $1              ;NO
0F89 06                CSA                     ;SEND START BIT
0F8A D4FD              ANI     %2              ;RESET READER RELAY
0F8C DC01              ORI     1
0F8E 07                CAS
0F8F C485      $2:     LDI     133             ;DELAY 1 BIT TIME
0F91 8F08              DLY     8
0F93 06                CSA                     ;GET BIT (SENSEB)
0F94 D420              ANI     020
0F96 9804              JZ      $3
0F98 C401              LDI     1
0F9A 9004              JMP     $4
0F9C C400      $3:     LDI     0
0F9E 9C00              JNZ     $4
0FA0 CAEA      $4:     ST      TEMP(P2)        ;SAVE BIT VALUE (0 OR 1)
0FA2 1F                RRL                     ;ROTATE INTO LINK
0FA3 01                XAE
0FA4 1D                SRL                     ;SHIFT INTO CHARACTER
0FA5 01                XAE                     ;RETURN CHAR TO E
0FA6 06                CSA                     ;ECHO BIT TO OUTPUT
0FA7 DC01              ORI     1
0FA9 E2EA              XOR     TEMP(P2)
0FAB 07                CAS
0FAC BAEB              DLD     NUM(P2)         ;DECREMENT BIT COUNT
0FAE 9CDF              JNZ     $2              ;LOOP UNTIL 0
0FB0 06                CSA                     ;SET STOP BIT
0FB1 D4FE              ANI     0FE
0FB3 07                CAS
0FB4 8F08              DLY     8               ;DELAY APPROX.  2 BIT TIMES
0FB6 40                LDE                     ;AC HAS INPUT CHARACTER
0FB7 D47F              ANI     07F
0FB9 01                XAE
0FBA 40                LDE
0FBB 3F                XPPC    P3              ;RETURN
0FBC 90B5              JMP     GECO

               ;*************************************
               ;*    PRINT CHARACTER AT TTY         *
               ;*************************************

0FBE 01        PUTC:   XAE
0FBF C4FF              LDI     255
0FC1 8F17              DLY     23
0FC3 06                CSA                     ;SET OUTPUT BIT TO LOGIC 0
0FC4 DC01              ORI     1               ; FOR START BIT.  (NOTE INVERS
0FC6 07                CAS
0FC7 C409              LDI     9               ;INITIALIZE BIT COUNT
0FC9 CAE8              ST      TEMP3(P2)
0FCB C48A      PUTC1:  LDI     138             ;DELAY 1 BIT TIME
0FCD 8F08              DLY     8
0FCF BAE8              DLD     TEMP3(P2)       ;DECREMENT BIT COUNT.
0FD1 9810              JZ      PUTC2
0FD3 40                LDE                     ;PREPARE NEXT BIT
0FD4 D401              ANI     1
0FD6 CAE9              ST      TEMP2(P2)
0FD8 01                XAE                     ;SHIFT DATA RIGHT 1 BIT
0FD9 1C                SR
0FDA 01                XAE
0FDB 06                CSA                     ;SET UP OUTPUT BIT
0FDC DC01              ORI     1
0FDE E2E9              XOR     TEMP2(P2)
0FE0 07                CAS                     ;PUT BIT TO TTY
0FE1 90E8              JMP     PUTC1
0FE3 06        PUTC2:  CSA                     ;SET STOP BIT
0FE4 D4FE              ANI     0FE
0FE6 07                CAS
0FE7 3F                XPPC    P3              ;RETURN
0FE8 90D4              JMP     PUTC
0000                    END     0
```

TELETYPE ROUTINES

ADD	0335	AESTK	1050	ANDOP	05F0	AT	0C96
BEGIN	0C54	BREAK	0288	CALBIT	0080	CALLML	0963
CHEAT	007C	CHEAT1	009B	CHPAGE	0BBA	CHRNUM	FFE7
CK1	0649	CKMODE	0644	CLEAR	0051	CLEAR1	0056
CLR	0C5C	CMP	0562	CMP1	05C2	CMP2	05CA
CMPR	05D9	COMMA	0DBD	DETPGE	0BC1	DFAULT	0C74
DIV	0410	DO	0CCD	DOLLAR	0D7A	DOLR1	0D8B
DOLR2	0D96	DONE	0135	DONE1	0143	DONE2	0144
DOPTR	FFFF	DOSTAK	107A	DOUBLE	0F1C	E0	0150
E0A	010D	E1	0195	E10	07C6	E11	081E
E12	086D	E12A	08E1	E13	0910	E14	0950
E15	097C	E16	09CC	E16A	0A2E	E17	0A4A
E18	0AA7	E19	0B04	E2	01CC	E3A	028A
E4	02DF	E5	030C	E6	0378	E6A	03D2
E8	064B	E8A	06E5	E8B	06B2	E9	074F
END	0E05	EQ	054C	EREG	FF80	ERR	0223
ERR1	0225	ERR2	0227	ERRNUM	0E2D	EVAL	07E6
EX1	0E66	EX2	0E69	EX3	0E6B	EX4	0E74
EX5	0E7D	EXECIL	0076	EXPR	0E5D	F1	0EB0

BRANCH TO PAGE 128

DIGI-KEY CORPORATION

Quality Electronic Components
Double-Digit Discounts
Save You Even More

2102-1	8080A	LM3909
$1.99	$24.95	69¢

NATIONAL SEMICONDUCTOR CLOCK MODULES

NATIONAL

MA1010A 12 Hour (AM-PM) Version . . . $13.00
MA1010C 24 Hour Version $13.00

MA1002A 12 Hour (AM-PM) Version $10.50
MA1002C 24 Hour Version. $10.50

0.5" High LED Digits

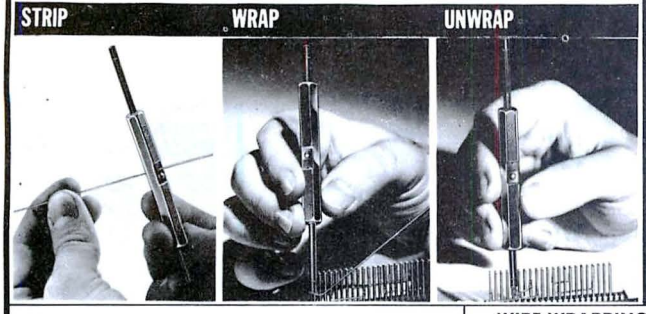
Special transformer and 6 switches, when ordered with module add $3.45.

0.84" High LED Digits

Special transformer and 6 switches when ordered with module add $3.45

Hobby-Wrap-30 $5.95

Strips, Wraps and Unwraps 30 ga. wire on standard wire wrapping pins



MA1002 & MA1010 Series Electronic Clock Modules

NATIONAL

The MA1002 & MA1010 Series Electronic Clock Modules are assembled and pretested modules which combine a monolithic MOS-LSI integrated clock circuit, 4-digit LED display, power supply and other associated discrete components on a single printed circuit board to form a complete electronic clock movement. The user need add only a transformer and switches to construct a digital clock for application in clock-radios, alarm or instrument panel clocks. Timekeeping may be from 50 or 60 Hz inputs and 12 or 24 hour display formats may be chosen. Direct LED drive eliminates RF interference. Time setting is made easy through use of "Fast" and "Slow" scanning controls.

The MA1002A and MA1010A have a 12 hour display with an AM and PM indicator. The MA1002C and MA1010C have a 24 hour display.

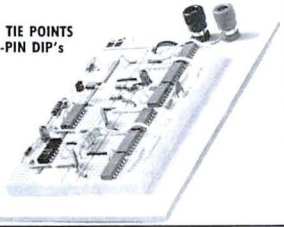
Features include alarm "on" and "PM" indicators, "sleep" and "snooze" timers and variable brightness control capability. The modules are extremely compact, the MA1002 measuring 1.375" by 3.05", the MA1010 measuring 1.75" by 3.75". This small size is achieved by bonding the I.C. to the back of the circuit board.

It is highly recommended that the transformer be obtained with the clock module as it is a special dual secondary type not otherwise readily available.

Double Digit Discounts Save You Even More!

ACE201-K
$24.95

1,032 SOLDERLESS PLUG-IN TIE POINTS
CAPACITY: *UP TO 12* **14-PIN DIP's**
TWO 5-way binding posts
Size: 4-9/16" by 7"
Kit form



WIRE WRAPPING WIRE

Red or Black Kynar insulated pre-cut and stripped 30 ga. wire. 1" of insulation is removed from each end.

Insulation Length	No. in Pkg.	Price per Pkg.
1.0"	100	$2.50
2.0"	100	$2.60
2.5"	100	$2.65
3.0"	100	$2.70
3.5"	100	$2.75
4.0"	100	$2.80
4.5"	100	$2.90
5.0"	100	$3.00
6.0"	50	$1.60
7.0"	50	$1.70
8.0"	50	$1.80
9.0"	50	$1.90
10.0"	50	$2.00
15.0"	50	$2.50

SLIDE SWITCHES

Single Pole Single Throw

.15	1.20/10	10.00/C

Double Pole Double Throw

.23	2.00/10	19.00/C

BATTERY HOLDERS

1-AA . . . 26c
2-AA . . . 42c
1-C 35c
2-C 42c
1-D 35c
2-D 42c

BISHOP GRAPHICS Printed Circuit Drafting

Aids are now available from Digi-Key

RADIAL ELECTROLYTICS

.47/50V. . .08	.65/10	22/50V. . . 12	1.00/10	330/25V . . 23	1.86/10
1/50V . . . 08	.65/10	100/6.3V . 09	.75/10	470/10R . . 21	1.71/10
2.2/50V. . .08	.65/10	100/10V . . 10	.77/10	470/16V . . 23	1.81/10
3.3/50V. . .08	.65/10	100/16V . . 11	.85/10	470/25V . . 29	2.35/10
4.7/35V. . .08	.65/10	100/25V . . 13	1.10/10	1000/10V . 24	1.96/10
4.7/50V. . .08	.68/10	100/50V . . 21	1.71/10	1000/16V . 29	2.35/10
10/16V . . .08	.65/10	220/10V . . 13	1.08/10	1000/25V . 42	3.33/10
10/25V . . .08	.65/10	220/16V . . 15	1.16/10	2200/10V . 42	3.33/10
10/50V . . .10	.75/10	220/25V . . 21	1.71/10	2200/16V . 54	4.30/10
22/16V . . .08	.67/10	220/50V . . 29	2.35/10	2200/25V . 58	4.67/10
22/25V . . .09	.70/10	330/10V . . 15	1.16/10	3300/16V . 89	7.14/10
		330/16V . . 21	1.66/10		

AXIAL ELECTROLYTICS

.47/10V . . 11	.90/10	33/25V. . . 14	1.15/10	330/16V . . 29	2.35/10
1/30V . . . 11	.90/10	33/50V. . . 19	1.52/10	330/25V . . 32	2.54/10
3.3/35V . . 12	.95/10	47/16V. . . 14	1.15/10	470/16V . . 32	2.55/10
3.3/50V . . 12	1.00/10	47/25V. . . 17	1.30/10	470/25V . . 37	3.00/10
4.7/25V . . 11	.90/10	47/50V. . . 21	1.17/10	1000/10V . 33	2.65/10
4.7/35V . . 12	.95/10	100/10V . . 14	1.13/10	1000/16V . 39	3.15/10
4.7/50V . . 12	1.00/10	100/16V . . 17	1.30/10	1000/25V . 56	4.50/10
10/25V. . . 12	1.00/10	100/25V . . 20	1.55/10	2200/10V . 50	3.96/10
10/50V. . . 14	1.15/10	100/50V . . 29	2.30/10	2200/16V . 62	4.95/10
22/16V. . . 12	1.00/10	220/10V . . 18	1.42/10	2200/25V . 79	6.36/10
22/25V. . . 13	1.05/10	220/16V . . 20	1.55/10	3300/16V . 95	7.63/10
22/50V. . . 17	1.32/10	220/25V . . 29	2.35/10	4700/16V1.09	8.70/10
33/16V. . . 12	1.00/10	220/50V . . 40	3.23/10	10000/10V 1.15	9.19/10
		330/10V . . 14	1.16/10		

INTEGRATED CIRCUITS — TTL, CMOS, LINEAR & MOS

7400 .21	7476 .32	74181 2.15	4012 .23	4520	1.14
7401 .21	7480 .70	74182 .79	4013 .40	4527	1.68
7402 .21	7482 .70	74184 2.19	4014 .96	4528	.88
7403 .21	7483 .70	74185 2.19	4015 .96	4585	1.23
7404 .21	7485 .89	74188 3.50	4016 .40	2102-1	1.99
7405 .21	7486 .28	74189 3.50	4017 1.05	8080A	24.95
7406 .25	7489 2.19	74190 1.23	4018 1.05	CA3046	.75
7407 .25	7490 .44	74191 1.23	4019 .23	LM211N	1.25
7408 .21	7491 .70	74192 .88	4020 1.14	LM309K	1.80
7409 .21	7492 .44	74193 .88	4021 1.14	LM324A	1.28
7410 .21	7493 .44	74194 .88	4022 .96	LM340T-5	1.25
7411 .21	7494 .70	74195 .88	4023 .23	LM340T-6	1.25
7412 .21	7495 .70	74196 .88	4024 .84	LM340T-8	1.25
7413 .25	7496 .70	74197 .88	4025 .23	LM340T-12	1.25
7414 .89	74100 1.28	74198 1.49	4026 1.68	LM340T-15	1.25
7416 .25	74107 .30	74199 1.49	4027 .40	LM340T-18	1.25
7417 .25	74109 .33	74251 1.09	4028 .89	LM340T-24	1.25
7420 .21	74121 .35	74279 .55	4029 1.14	LM3900N	.88
7421 .25	74122 .44	74365 .67	4030 .23	LM3909N	.69
7423 .35	74123 .61	74366 .67	4033 1.51	MC1456V	1.00
7425 .35	74125 .40	74367 .67	4034 3.50	MC1458V	.53
7426 .25	74126 .40	74368 .67	4035 1.14	MC3302P	1.15
7427 .33	74132 .70	8093 .40	4040 1.14	NE536T	3.24
7428 .28	74141 .88	8094 .40	4041 .79	NE540L	2.04
7430 .21	74145 .70	9095 .67	4042 .79	NE555V	.48
7432 .25	74147 1.63	8096 .67	4043 .70	NE556A	.88
7433 .30	74148 1.30	8097 .67	4044 .70	NE560B	3.83
7437 .25	74150 1.16	8098 .67	4046 1.86	NE561B	3.83
7438 .25	74151 .70	75150 1.16	4049 .40	NE562B	3.83
7440 .21	74153 .65	75450 .88	4050 .40	NE565A	1.25
7442 .53	74154 1.03	75451 .61	4051 1.26	NE566V	1.28
7443 .63	74155 .70	75452 .61	4052 1.26	NE567V	1.36
7445 .70	74156 .70	75453 .61	4053 1.26	uA709CV	.44
7446 .70	74157 .70	75454 .61	4060 1.58	uA710CA	.44
7447 .70	74160 .88	75491 .81	4066 .79	uA711CA	.53
7448 .70	74161 .88	75492 .84	4071 .23	uA723CA	.60
7450 .21	74162 .88	75493 1.09	4072 .23	uA733CA	.75
7451 .21	74163 .88	75494 1.19	4073 .23	uA741CV	.44
7453 .21	74164 .96	82525 2.19	4075 .23	uA747CA	.70
7454 .21	74165 1.15	4000 .23	4081 .23	uA748CV	.49
7459 .21	74166 1.26	4001 .23	4082 .23	uA7805CU	1.25
7460 .21	74170 2.64	4002 .23	4502 .79	uA7806CU	1.25
7470 .30	74173 1.42	4006 1.23	4510 1.14	uA7808CU	1.25
7472 .30	74174 .98	4007 .23	4511 1.05	uA7812CU	1.25
7473 .30	74175 .93	4008 .79	4514 2.80	uA7815CU	1.25
7474 .30	74176 .79	4009 .44	4515 2.80	uA7818CU	1.25
7475 .49	74177 .79	4010 .44	4516 1.23	uA7824CU	1.25
	74180 .70	4011 .23	4518 1.14		

SILICON TRANSISTORS

MPS918, MPS930, MPS2222A, MPS2369A, MPS2712, MPS2907A, MPS3392, MPS3393, MPS3394, MPS3395, MPS3563, MPS3565, MPS3638, MPS3638A, MPS3640, MPS3641, MPS3643, MPS3645, MPS3646, 2N3904, 2N3906, 2N4124, 2N4126, 2N4401, 2N4403, 2N4410, PN4888, 2N5087, 2N5089, PN5129, PN5133, PN5134, PN5137, PN5138, PN5139, 2N5210, PN5964 . . . . 16, $1.55/10, $13.60/100 of same part no.

MPF102 .36 $30.60/C 2N5457 .48 $41.00/C MPSA13 .28 $24.00/C 2N3055 .99 $85.00/C

WIRE WRAPPING WIRE IN BULK

Red or Black 30 ga. Kynar

100' $2.00 500' $8.50 1000' $15.00

ENTIRE ROCKWELL CALCULATOR LINE ON DISCOUNT IN OUR LATEST CATALOG

MOLEX PINS

Make your own IC sockets on PC board

.85/C	8.20/M	38.20/5M	275.00/50M

SILICON DIODES

1N4001	.64/10	5.50/C	$49/M
1N4002	.66/10	5.60/C	$51/M
1N4003	.68/10	5.80/C	$52/M
1N4004	.70/10	5.95/C	$54/M
1N4005	.82/10	7.05/C	$63/M
1N4006	.90/10	7.75/C	$69/M
1N4007	.99/10	8.60/C	$77/M
1N4148	.40/10	3.50/C	$29/M

Double Digit Discounts Save You Even More!

RESISTOR ASSORTMENTS

5 ea all 10% ¼ w. val. 10 thru 5.6 meg (350 pcs)	$12.00
5 ea all 10% ½ w. val. 2.2 thru 22 meg (425 pcs)	$12.00

½ WATT ZENER DIODES

1N5226B 3.3v .15 $11/C	1N5236B 7.5v .15 $11/C
1N5227B 3.6v .15 $11/C	1N5237B 8.2v .15 $11/C
1N5228B 3.9v .15 $11/C	1N5238B 8.7v .15 $11/C
1N5229B 4.3v .15 $11/C	1N5239B 9.1v .15 $11/C
1N5230B 4.7v .15 $11/C	1N5240B 10v .15 $11/C
1N5231B 5.1v .15 $11/C	1N5241B 11v .15 $11/C
1N5232B 5.6v .15 $11/C	1N5242B 12v .15 $11/C
1N5233B 6.0v .15 $11/C	1N5243B 13v .15 $11/C
1N5234B 6.2v .15 $11/C	1N5244B 14v .15 $11/C
1N5235V 6.8v .15 $11/C	1N5245B 15v .15 $11/C

PERF BOARD

0.1" spacing between holes

4½"x .6"	$1.71
4½"x17"	$4.25

TERMINALS

for above perf board

$1.50/C

LED LAMPS

NSL5053 T-1 ¾ .18 $15/C
NSL5056 T-1 ¾ .18 $15/C

LED DISPLAYS

FND357 CC .375" . . $1.75
FND500 CC .500" . . $1.75
FND 507 CA .500 . . $1.75
FND800 CC .800" . . $3.50
FND807 CA .800" . . $3.50

DATA BOOKS

DATA BOOKS

TTL IC's 595 p.	$4.00
Linear IC's 957 p.	$5.00
CMOS IC's 256 p.	$3.00
Transistors 288 p.	$3.00
Memory IC's 592 p.	$3.00
Interface IC's 464 p.	$4.00
Volt. Regs. 128 p.	$3.00
Linear Appl. 1 432 p.	$4.00
Linear Appl. 2 246 p.	$3.00
Audio 196 p.	$3.00

WIRE-WRAPPING TOOL

$5.95

Wraps, Unwraps & Strips 30 ga. Wire

¼ & ½ WATT 5% CARBON FILM RESISTORS

5¢ each in multiples of 5 per value
$1.70/100 & $12.00/1000 of same value

1 ohm thru 1.0 megohm

I.C. SOCKETS

8 Pin Solder	.17	1.60/10
14 Pin Solder	.20	1.90/10
16 Pin Solder	.22	2.10/10
18 Pin Solder	.29	2.75/10
24 Pin Solder	.38	3.60/10
28 Pin Solder	.45	4.25/10
40 Pin Solder	.63	6.00/10
8 Pin W-W	.24	2.30/10
14 Pin W-W	.26	2.50/10
16 Pin W-W	.30	2.85/10
18 Pin W-W	.60	5.70/10
24 Pin W-W	.96	9.10/10
28 Pin W-W	1.12	10.00/10
40 Pin W-W	.92	8.75/10

HARDWARE

2-56 1/4 Screw	.99/C	7.20/M
2-56 1/2 Screw	.99/C	7.65/M
4-40 1/4 Screw	.55/C	3.60/M
4-40 1/2 Screw	.60/C	4.05/M
6-32 1/4 Screw	.65/C	4.40/M
6-32 1/2 Screw	.75/C	4.85/M
8-32 3/8 Screw	.90/C	5.85/M
8-32 5/8 Screw	.99/C	7.00/M
2-56 Hex Nut	.55/C	3.60/M
4-40 Hex Nut	.55/C	3.75/M
6-32 Hex Nut	.60/C	4.00/M
8-32 Hex Nut	.60/C	4.15/M
No. 2 Lockwasher	.85/C	5.75/M
No. 4 Lockwasher	.45/C	3.00/M
No. 6 Lockwasher	.45/C	3.00/M
No. 8 Lockwasher	.45/C	3.00/M

DOUBLE-DIGIT DISCOUNT SCHEDULE

Merchandise	Discount	Merchandise	Discount
$ 0.00-$24.99	NET	$100.00-$499.99	Less 15%
		$500.00-$999.99	Less 20%
$25.99-$99.99	Less 10%	$1000.00 & Up	Less 25%

Then Add the Standard Charge Below

STANDARD SHIPPING/HANDLING CHARGE

If your merchandise total after discount is between

$ 0.00-$ 4.99 . . . add $2.00	$ 50.00-$99.99 . . . add $0.25
$ 5.00-$24.99 . . . add $0.75	$100.00 & up . . . No Charge
$25.00-$49.99 . . . add $0.50	

Includes shipping and insurance to USA & Canada

COD ORDERS ACCEPTED FOR SAME DAY SHIPMENT — CALL 218-681-6674

Send for free catalog or mail reader's service card

"Only Quality Components Sold!"
DON'T FORGET TO APPLY DISCOUNT SCHEDULE

DIGI-KEY CORPORATION

Box 677, Thief River Falls, MN 56701 218-681-6674

MICRO-MARKET

COMPUTER HOBBYISTS!
Nationwide Classified Advertising Newsletter
Buy & Sell Hardware & Software
new/used/unique/low-cost
$3.75 for 18 issues.
Free sample issue on request.
ON_LINE Newsletter
24695 Santa Cruz Hwy, Los Gatos, Ca. 95030

Computer Mart of New York Inc.
314 Fifth Ave. ★ New York, N.Y. 10001
Microcomputers, Books, Components, Parts
Authorized Dealer—Service—Friendly Advice
Open 10-6 PM Mon-Sat *STAN VEIT,*
10-9:30 PM Thurs Eve. *Storekeeper*
Telephone (212) 279-1048

LONG ISLAND
IMSAI Dealer
COMPUTER MICROSYSTEMS
Kits • Parts • Service • Books • Magazines
1309 Northern Blvd., Manhasset, N.Y. 11030
Telephone (516) 627-3640

★ 1702A PROM PROGRAMMING SERVICE ★
EPROM copy – $1 Hex form input – $3 Prime 1702A – $9. Software available including: Program loaders, Memory checkers, Video drivers and Code convertors.
We offer: Fast turnaround, free coding forms, quantity discount. Write for free catalogue.
MICROTRONICS
P.O. Box 7454F Menlo Park, CA 94025

THE COMPUTER CORNER
"Brains are our Business"
✓ IMSAI 8080, POLY-88, Processor Tech
✓ Books, Magazines, TTY Supplies, Brain Games
Hours: 10-6 pm daily & Sat.; Thurs. 10-9 pm
White Plains Mall, 200 Hamilton Ave.
WHITE PLAINS, NY 10601
Telephone: (914) 949-DATA

DISCOUNTS!
5% savings on single purchases. No shipping charges, no tax! Many popular products. Write for free information.
MONTANA COMPUTER CENTER, INC.
2512 Grand Ave., Billings, Montana 59102

A.D.M.
Communications, Inc.
Teletype & Communications Equipment
Buy—Sell—Recondition
1322 Industrial Ave.
Escondido, CA 92025
(714) 747-0374 Telex. 695097

TELETYPES $800
Model 33 ASR • Newly refurbished
RS232 • full/half duplex
30 day warranty
THE MULL COMPANY
920 Laurel Avenue
Menlo Park, CA 94025
(415) 327-7509

SERIES 33 TTY's FOR SALE OR RENT
EXCLUSIVE VARDON STYLING REDUCES OPERATING NOISE 50% OR MORE

		SALE	RENTAL
ASR	VARDON STYLING	1136.00	45.50/mo
	STANDARD STYLING	940.00	40.50/mo
KSR	VARDON STYLING	875.00	40.50/mo
	STANDARD STYLING	650.00	30.50/mo

OPTIONAL SPECIALS

w/Acoustic coupler	add $6/mo
W/Hard wire modem (2 wire)	add $6/mo
W/EIA RS-232	add $3/mo
W/2 E.C. or Nova interface	add $4/mo
W/DDD Originate-Auto Answer modem	add $23/mo
W/TWX/DDD modem	add $25/mo

TERMS
1. Prices guaranteed for term of lease
2. All equipment F.O.B. Dallas, Texas
3. All equipment unconditionally guaranteed for 30 days
4. Customer must provide maintenance after 30-day warranty period

GET COMPLETE DETAILS WITH A DIRECT CALL:
(214) 252-7502 • TWX 910-860-5761 • TELEX 73-0022
vardon & associates, inc.
930 N. Beltline Road ■ Irving, Texas 75061
A subsidiary of Pioneer Texas Corporation

MARKET YOUR SOFTWARE
CALL OR WRITE
Microcomputer Software Depository
2361 E. Foothill Boulevard
Pasadena, California 91107
(213) 449-0616

ELIMINATE INTERFERENCE!
Compact power line filter keeps line interference out of your system. Bonus! Keep your system noise from interfering with neighborhood TV, FM and AM receivers. Wire directly into Teletype, Computer, Terminal or Power Supply. Complete installation instructions. ☆*Only $8.95 Postpaid.*
Electronic Specialists, Inc.
171 So. Main, Natick, Mass. 01760
(617) 655-1532

AMCO HAS APPLE!
APPLE COMPUTERS
AMCO ELECTRONIC SUPPLY
635 E. ARROW HIGHWAY
AZUSA, CA 91702
Phone: (213) 332-2216

TEN LOW CAPITAL COMPUTER BUSINESSES
to start at home part-time. 50 page report loaded with details never published. $5.00 (Guaranteed Refundable) to ***DATASEARCH***, 730 Waukegan, St 108, Dept. F, Deerfield IL 60015.

MICRO-MARKET AD RATE: $25.00 per column inch. Max. 4 column inches per ad. Submit ads to Micro-Market Ads—INTERFACE Magazine, P.O. Box 1234, Cerritos, CA 90701.

SOUTH FLORIDA
Sunny Computer Stores, Inc.
Authorized Dealer:
IMSAI, Southwest, Cromemco, CSC, Vector and others
Books, Magazines, Newspapers, Service and Supplies
University Shopping Center
1238A South Dixie Highway
Coral Gables, Fla. 33146
(305) 661-6042

End your video display problems quickly with a "PIXE-VERTER"
A transistorized modulated oscillator which instantly converts a TV receiver into a top notch video monitor. Connects directly to antenna terminals. No wiring modifications required to the set. Operates on channels 2-6. Power requirements 5-6 volts . 3 ma. Ideally suited for computers, electronic games, TV cameras, video recorders, etc. Size: 1.25" x 2.1". Kit model #PXV-2A $8.50. Order by phone or mail.
DIAL 402-987-3771
ATV Research 13-I BROADWAY DAKOTA CITY, NEBR 68731

COMPUTER HOBBYISTS
Protect your valuable I C's
The UN ⚡ ZAP Kit
$14.95 POSTPAID: U.S., Can. & Mex.
Other Countries add $2 postage & handling
Kit includes 27"x36" pad and 6 ft. wriststrap of Heavy Duty conductive plastic
Larger sizes and quantity discounts available. No COD's.
Upland COMPUTER Labs
P. O. Box 1365
75 E. Ninth St., Upland, Calif. 91786
714 / 981-1503
A MICROCOMPUTER STORE

INTRODUCTION TO MICROCOMPUTERS
VOLUME 2 IS HERE!
$8.50 gets you a copy fast.
($8.50 INCLUDES postage and handling)
MONTANA COMPUTER CENTER
2512 Grand Avenue, Billings, Mt. 59102

HOLIDAY SPECIAL$
Edge Connector and Guides for IMSAI 8080
$3.95 each
Add $2.00 per order for shipping and handling
NY state residents add sales tax
Write for our super prices on IMSAI, TDL, Polymorphic and National Multiplex
computer enterprises
CALL: (315) 637-6208
P.O.Box 71
Fayetteville, N.Y. 13066

FIFO FLEA MARKET

HP-45 CRYSTALS: I have some crystals to make the timebase in an HP-45 accurate so it can be used as a 10 event stopwatch. I am willing to sell these for $13 and will include some instructions on how to install it. Mark S. Egbert, 1514 S. 320 E. Orem, UT 84057. (801)224-2102.

WANT SYSTEM, have R/C Gold Mine worth over $3300 to TRADE. Helicopters, planes, radios, cars, engines, field boxes and more. Interested in ALTAIR 8800B or IMSAI 8080 System in trade (will accept cash). If you're ready to move outdoors with your hobby please call A.B. Clark, (305)238-3408 nights, or (305)873-2528 days. Miami, FL.

SELECTRIC: One Dura Mach 10 wide platten Selectric based terminal with dual paper tape punches and readers. Computer interface available. IBM serviced and adjusted typer. Reconditioned and tested. Contact Ron Jenkins, 4490 Sirius Ave., Lompoc, CA. (805)733-2575 evenings except Mon. and Wed.

FOR SALE: Two MITS 4K static memory boards. Fully essembled and 100% operational. I am switching to 16K static boards. $160 each. David Milhouse, 2823 Griffa Ave., Columbus, IN 47201.

MICRODATA REALITY: Are there any other computer hobbyists using this system? If so, I'd like to say hello, swap notes and programs, etc. Would also like to know where to buy a 4- or 8-way video terminal interface card and other peripherals for Microdata Reality (Model 1600 cpu). Jack Hardman, 140 Forest Ave., Glen Ridge, NJ 07028. (201)429-8880.

FOR SALE: (1) Mil Mod-8 backplace w/edge connectors, 1702 programmer - w/most parts $40. (1) Digital Group cassette tape controller A&T $150. (2) Phi-deck tape drives $115 ea. (1) Solid state Music 4K static RAM, no 2102's, $60. (2) Control Data Acoustical Couplers RS-232 Interface, $12.50 ea. Please include sufficient postage, thank you. Pam Raines, 325 Doty, Ann Arbor, MI 48104.

FOR SALE: IMSAI 8080 22 slot Mother Board. Assembled and tested. All software rights. $775. Contact: James T. Burnham, 4444 Hansen Avenue, Apt. 104, Fremont, CA 94536. (415) 793-7971.

STARTING New Correspondence Exchange Club, would appreciate response from you on any useful project with a computer, visual or graphic, video camera interfacing to a computer. Any computer/printer generated pictures or posters; STARTREK, Playboy, Snoopy, Cat, any figure or design on paper. Joe Penora, 920 First Ave., Franklin Square, NY 11010.

WANTED: Centronics (101, 102) Diablo, Datapoint (2200, 1100), or H.W., or H.P., equivalent. NCR, Entrex, Univac, tubes, printers, disk, diskette drives. Wango, Wang, Kennedy, Ampex, Pertec PEC tape drives, in any condition, working units, scrap frames or spare parts. Software Datapoint or H.W., H.P., equivalent. Pat Snyder, Box 31, Franklin Square, NY 11010.

FOR SALE: 3 MITS 88-4mcds (4K Static) at $150 each, 2 MITS 88-4mcd (4K dyn.) at $190 each, 2 MITS IK Static at $75 each. All socketed, all work (together even) all less than 40 hrs total time. Guaranteed not assembled by a graduate engineer. JEDJ 2127 Harper, Lawrence, KS 66044. We're going from 8 to 16 bit machine!

FOR SALE: Altair 8800 - 8K memory, RS-232 serial interface, cassette interface with motor control, fan, extra edge connectors installed, 8K BASIC and games, etc. Cost of kit was over $1200 - will sell for $895 - assembled and working perfectly. Documentation and MITS newsletters included. Don Barrett, 284 Aspenwood Lane, Encinitas, CA 92024. (714)436-9060.

PDP-8 MINI COMPUTER (Straight-8) for sale or trade. Mounted in attractive walnut cabinet (wife approved for den). Complete with maintenance manual, drawings, 4K Basic, 4K Focal, Editor, PAL III assembler, Disassembler, TTY interface, games, etc., etc. Perfect condition. Free DECUS membership available for additional software. Presently in use with ASR Model 33 teletype. Can be sold with or without teletype. Sale price w/o teletype $800.00. Rex Eagle, P.O. Box 10498, Charlotte, N.C. 28288. (704) 374-6557.

FOR SALE: TTY Paper Tape. 1 inch wide, 8 inch diameter rolls, buff, oiled. Perfect for ASR-33. $5.00 + UPS shipping per 7 roll carton (12 lbs) or $15.00 per 28 roll case + UPS. Two or more cases, $12.50 each + UPS. D. S. Parker, 1007 3rd St. No. 3, Davis, CA 95616. (916) 758-2341 after 5:00 p.m.

I WILL BUY your new, used, or dead Altair 8800, working or not, for cash. Am desperate for Altair; help out a poor college kid. Will also trade for your machine; have a scope and a teletype. Call anytime. Jim Webb, P.O. Box 5224, Carson, CA 90749. (213) 325-4684.

FOR SALE. All completely assembled and fully tested. ALTAIR 8800 with 5K RAM, PT 16-slot mother, MITS 4K BASIC on paper tape, fan; best offer over $700 takes it. SWTP CT-1024 with screen read, manual and computer cursor boards, 110-150-300-600-1200 baud SI/O, video monitor; best offer over $250. I'll allow $325 for your clean Micro-Term ACT-1 if you take it all. Bob Ripley, Route 3, Fayetteville, AR 72701. (501) 267-3172 after 5 CDT.

FIFO FLEA MARKET ADS

★ PERSONALS FREE ★

Max. 10 lines per ad. Submit Ads to: FIFO Ads—**INTERFACE AGE** Magazine, P.O. Box 1234, Cerritos, CA 90701.

WANTED: Need Varian 620L or L-100, Contact Frank Jacobson, (714) 842-6439.

VIATRON SYSTEM 21 Boards No. 1 thru 7 Complete with LSI and only three known problems. (over 80% AOK) $125.00 Viatron Key Board $25.00. Viatron power supply $25.00. 8 level Friden paper tape punch $50.00. Paper tape reader $45.00 (both for $75.00). WB5BLX, 3728 Wilkie Way, Ft. Worth, TX 76133.

WANTED: A Hi-spd. computer terminal and or coupler (modem). Preferably a CRT. At least 12 lines and at least a 64 character line, preferably 80. ASCII. Full/half duplex coupler. Both CRT and coupler must be capable of 30 C.P.S. Under $800 for terminal. Write to: Mark Chapman, 7015 Tupa Dr., Edina, Mn. 55435.

HELP — anyone have info. on general computer terminal PT2900000. These were sold by ALTAJ several years ago. Need input/output info. J.P. Chalala, RD 1, Willow Street, PA 17584.

CARD READER with documentation $100.00 or trade. Bob Stevens, 2361 E. Foothill Blvd., Pasadena, CA 91107. (213) 449-0616.

FOR SALE OR TRADE: Two signetics 3000-series Bit Slice 8-bit development kits, *each* containing 1-3001 MCU, 4-3002 CPE, 1-74S182, 1-8T31, 2-8T26A, and 3-82S114 256x8 PROM. Normally sells for $100/ea. Information on a PDP-11 emulator using these chips is available. S.L. Diamond, P.O. Box 22428, San Francisco, CA 94122.

FOR SALE: CRYPTOGRAPHIC PROGRAM that will baffle the codebreakers of the CIA. ENCODE and/or DECODE your private correspondence for maximum security. Documentation of cypher technique, program listing and punch paper tape in BASIC for only $6. JON STEDMAN, 1528 Summit Rd., Berkeley, CA 94708.

SWAP OR SELL: SWTP CT-1024 TVT II with Serial I/O, Manual Cursor, Screen Read, P/S. ALTAIR 8080 CPU board. Tarbell Cassette interface. HAL ST-6 Teletype converter with all options (Can be used as a Modem). All in like new condition and guaranteed perfect. Need Hard-copy printer or ??? Michael Shrayer, 930 So. Bonnie Brae

VECTORED FROM PAGE 124

```
F10     0F12    F2      0EB6    F3      0EBD    F4      0EC7
F5      0ED0    F6      0ED8    F7      0EE5    F8      0EF0
F9      0EFD    FACTOR  0EA8    FAIL    05E1    FAILHI  FFEC
FAILLO  FFED    FALSE   05C8    FIN     02B2    FNDLBL  0BDE
FNDPOE  0B94    FOR     0D26    FOR1    0D46    FOR2    0D48
FORPTR  FFFE    FORSTK  108A    GECO    0F73    GE0     0560
GE01    05BE    GETL    0753    G01     0CF2    GOSUB   0CE7
GOTO    0CD9    GTR     055C    GTR1    05B3    GTROP   0E5B
HEX     0654    HI      FFEE    HILINE  FFF7    IF      0CA6
IF1     0CD2    IGNORE  09B5    ILC1    00AA    ILCALL  00A0
IN1     0DDE    IN2     0DF2    IN3     0E01    IND     0534
INPUT   0DCD    INSRT   0820    ISTRNG  0B22    JMPBIT  0040
LABLHI  FFF2    LABLLO  FFF3    LBUF    10D6    LE0     0558
LE01    05AA    LET     0C87    LIST    0C31    LIST1   0C43
LIST2   0C45    LIST3   0C47    LISTNG  FFF5    LIT1    0A30
LO      FFEF    LOLINE  FFF8    LSS     0554    LSS1    05A2
LST     02E1    LST2    02FF    LST3    030E    LST4    0314
LST5    0324    LSTK    FFFD    MESGS   0F37    ML      0E0E
MODULO  09BC    MOVE    0804    MOVESR  0949    MOVSTR  0B52
MUL     037A    NEG     0363    NE0     0550    NE01    0599
NEW     0C69    NEW1    0C76    NEWPGM  0BCD    NEXT    0D0C
NEXTV   0A85    NEXTV1  0AEA    NLINE   0215    NOJUMP  009D
NOTOP   05F8    NUM     FFEB    NUPAGE  0B89    NXT     028C
NXT1    02A9    OROP    05F4    P1      0001    P1HIGH  FFF0
P1LOW   FFF1    P2      0002    P3      0003    PAGE    FFF6
PCHIGH  FFFA    PCLOW   FFFB    PCSTAK  10A6    PCSTK   FFF9
PGE     0D63    PGM     1120    PR0AE   0912    PR1     0DA3
PR2     0DAA    PR3     0DB9    PR4     0DC2    PR5     0DC7
PR6     0DC9    PRINT   0D9A    PRMPT1  0C21    PRN     0197
PRN1    01CE    PRNUM   0F2B    PRNUM1  0F2F    PROMPT  0C1A
PRS     017E    PRS1    0193    PSTRNG  0B06    PUTC    0FBE
PUTC1   0FCB    PUTC2   0FE3    PUTPGE  0B77    PUTSTR  0B32
RANDOM  090E    REL1    0E3A    REL2    0E44    REL3    0E4B
REL4    0E4F    REL5    0E59    RELEXP  0E31    REM     0E22
RETEXP  0E87    RETURN  0CFC    RNDF    FFE6    RNDX    FFE5
RNDY    FFE4    RSTR    0148    RSTR1   0152    RSTR2   0167
RTN     00FB    RUN     0C4D    RUNM0D  FFF4    SAV     010F
SAV1    012B    SAV2    0131    SAVEDO  0974    SAVFOR  0A42
SBRPTR  FFFC    SBRSTK  106A    SETZ    058D    START   0C18
STAT    0D50    STATUS  0952    STMT    0C82    STORE   04C9
STRT    02C8    SUB     034C    SYNTAX  0E2B    T1      0E8B
T2      0E94    T3      0E9D    TEMP    FFEA    TEMP2   FFE9
TEMP3   FFE8    TERM    0E89    TOP     0990    TST     00C5
TSTBIT  0020    TSTNUM  06B4    TSTVAR  04E9    UNT     0CB8
UNTIL   0924    VARS    101C    X0      00EC    X1      0165
X10     04E2    X11     054A    X12     0597    X12A    05EE
X12B    0637    X12C    067A    X13     06E3    X14     0751
X15     07B4    X16     07E4    X17     081C    X19     086B
X19A    08E5    X20     0909    X21     094E    X22     098E
X23     09CA    X24     0A2C    X25     0A83    X26     0AE8
X27     0B30    X4      01CA    X5      0221    X5A     0286
X6      02DD    X6A     030A    X7      034A    X8      0376
X9      03EF    X9A     0439    X9B     04A4    XCHGP1  0639
XFER    0171    XFER1   0179    ZZ0001  101C    ZZ0002  1120
ZZ0003  0FBD    ZZ0004  0FBD    ZZ0005  0FBD    ZZ0006  0FBD
ZZ0007  0F37    ZZ0008  0F37    ZZ0009  0FBD    ZZ000A  10D6
ZZ000B  0FBD    ZZ000C  101C    ZZ000D  0FBD    ZZ000E  0002
ZZ000F  0002    ZZ0010  0006    ZZ0011  0002    ZZ0012  0003
ZZ0013  0002    ZZ0014  0002    ZZ0015  0002    ZZ0016  0002
ZZ0017  0005    ZZ0018  0004    ZZ0019  0002    ZZ001A  0007
ZZ001B  0004    ZZ001C  0004    ZZ001D  0003    ZZ001E  0002
ZZ001F  0006    ZZ0020  0005    ZZ0021  0002    ZZ0022  0003
ZZ0023  0006    ZZ0024  0005    ZZ0025  0002    ZZ0026  0003
ZZ0027  0005    ZZ0028  0002    ZZ0029  0002    ZZ002A  0005
ZZ002B  0003    ZZ002C  0003    ZZ002D  0007    ZZ002E  0003
ZZ002F  0004    ZZ0030  0003    ZZ0031  0002    ZZ0032  0005
ZZ0033  0002    ZZ0034  0003    ZZ0035  0002    ZZ0036  0003
ZZ0037  0003    ZZ0038  0002    ZZ0039  0006    ZZ003A  0003
ZZ003B  0003    ZZ003C  0007    ZZ003D  0003    ZZ003E  0002
ZZ003F  0002    ZZ0040  0002    ZZ0041  0002    ZZ0042  0002
ZZ0043  0002    ZZ0044  0002    ZZ0045  0002    ZZ0046  0002
ZZ0047  0002    ZZ0048  0002    ZZ0049  0002    ZZ004A  0002
ZZ004B  0002    ZZ004C  0002    ZZ004D  0002    ZZ004E  0003
ZZ004F  0002    ZZ0050  0003    ZZ0051  0002    ZZ0052  0003
ZZ0053  0003    ZZ0054  0003    ZZ0055  0004    ZZ0056  0004
ZZ0057  0008    ZZ0058  0003    ZZ0059  0002    ZZ005A  0003
ZZ005B  0005    $0      002A    $0      0420    $0      076F
$0      0996    $0      0B91    $1      0043    $1      01C6
$1      023D    $1      0397    $1      0443    $1      05FA
$1      06E7    $1      0772    $1      0838    $1      0930
$1      097E    $1      099C    $1      09FD    $1      0A4C
$1      0A8F    $1      0B12    $1      0B46    $1      0BA3
$1      0BE4    $1      0F7B    $10     0AA9    $2      01FF
$2      025A    $2      03A8    $2      0454    $2      0707
$2      07B2 *  $2      0842    $2      09A3    $2      0ADD
$2      0B28    $2      0BFC    $2      0F8F    $3      0261
$3      03D4    $3      04AC    $3      084A    $3      0ADF
$3      0C11    $3      0F9C    $4      03F1    $4      0857
$4      0FA0    $5      085E    $ABOR   06C6    $ADD    08E3
$ADD1   08FF    $CALB   8000    $CR     07D8    $DOWN   0885
$END    04C3    $END    06AC    $END    0B4C    $ENT1   04A6
$ENTE   068B    $ENTE   07B6    $EXIT   0402    $FAIL   04FB
$JMPB   4000    $LETR   067C    $LOOP   002C    $LOOP   00DB
$LOOP   0203    $LOOP   0241    $LOOP   03B6    $LOOP   0460
$LOOP   066C    $LOOP   06F7    $LOOP   09D9    $LOOP   0AD1
$LOOP   0B38    $LOOP   0B68    $LOOP   0BAA    $MAYB   050D
$MOVE   086F    $MSG    0247    $NE0    00EE    $NOT    0628
$OK     051A    $OK     0688    $OR     0616    $POS    043B
$PRNT   01EF    $REDO   093C    $REDO   0AF6    $RET    06D6
$RUB    07CE    $SCAN   00C7    $SHIF   0692    $SHIF   0714
$SKIP   0664    $TSTB   2000    $UP     0895    $UP1    08A1
$UP2    08C2    $UP3    08C4    $UP4    08D4    $XH     07C8
$XU     07A6
```

```
NO ERROR LINES
SOURCE CHECKSUM = 33FE
INPUT FILE     1:NIBL1127.SRC
```

ADVERTISING INDEX

CAST YOUR VOTE

The authors in this issue would like to know your feeling as to which article you consider the best. Cast your vote for one, two or all and grade them on a point scale 1 (lowest) to 10 (highest). At the end of the voting period all votes will be tallied, the highest scoring article announced and the author will receive $100.00 bonus. Use only the enclosed "Voting and Reader Service Card" for your ballot.

READER SERVICE CARD

January 1977 Issue — Void after March 31, 1977

(Please type or print)

Name _____ Title _____

Company _____ Phone (A/C) _____

Address _____ ☐ Home ☐ Business

City _____ State _____ Country _____ Zip _____

Please send information on items circled below.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270
271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330
331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360
361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390

My primary interests are in ☐ Hardware ☐ Software.

I am a ☐ Hobbyist ☐ Professional Engineer ☐ Businessman.

INTERFACE™

January 1977 Issue — Void after March 31, 1977

VOTING CARD

(Please type or print)

Name _____ Title _____

Company _____ Phone (A/C) _____

Address _____ ☐ Home ☐ Business

City _____ State _____ Country _____ Zip _____

Please send information on items circled below.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270
271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330
331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360
361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390

My primary interests are in ☐ Hardware ☐ Software.

I am a ☐ Hobbyist ☐ Professional Engineer ☐ Businessman.

INTERFACE™

January 1977 Issue — Void after March 31, 1977

(Please type or print)

Name _____ Title _____

Company _____ Phone (A/C) _____

Address _____ ☐ Home ☐ Business

City _____ State _____ Country _____ Zip _____

Please send information on items circled below.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270
271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330
331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360
361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390

My primary interests are in ☐ Hardware ☐ Software.

I am a ☐ Hobbyist ☐ Professional Engineer ☐ Businessman.

SUBSCRIPTION APPLICATION FORM

PAYMENT MUST ACCOMPANY FORM

SUBSCRIPTION FEE per year: $10.00 U.S., $12.00 Canada/Mexico, $18.00 Foreign.

Make check payable to: INTERFACE™ Magazine

P. O. Box 1234, CERRITOS, CA 90701

Name _____ Title _____

Company _____

Home Address _____

City _____ State _____ Country _____ Zip _____

Date _____ Signature _____

SUBSCRIPTION APPLICATION FORM

PAYMENT MUST ACCOMPANY FORM

SUBSCRIPTION FEE per year: $10.00 U.S., $12.00 Canada/Mexic

$18.00 Foreign.

Make check payable to: INTERFACE™ Magazine

P.O. Box 1234, Cerritos, CA 90701

Name _____ Title _____

Company _____

Home Address _____

City _____ State _____ Country _____ Zip _____

Date _____ Signature _____

FIRST CLASS
PERMIT No. 18
Cerritos, Ca. 90701

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Interface Hotline
P.O. Box 1234
Cerritos, Ca. 90701

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Interface Hotline
P.O. Box 1234
Cerritos, Ca. 90701

FIRST CLASS
PERMIT No. 18
Cerritos, Ca. 90701

PLANNING TO MOVE?

Let us know 8 weeks in advance so that you won't miss a single issue of INTERFACE.

Attach old label where indicated and print new address in space provided. Also include your mailing label whenever you write concerning your subscription. It helps us serve you promptly.

If you have no label handy, print OLD address here.

Please Print

Name

Company

Address

City State Zip code

Affix Label **Print NEW address here**

Name

Company

Address

City State Zip code

MAIL COMPLETED CARD TO:

INTERFACE™

P. O. Box 1234
Cerritos, California 90701

Let us know 8 weeks in advance so that you won't miss a single issue of INTERFACE.

Attach old label where indicated and print new address in space provided. Also include your mailing label whenever you write concerning your subscription. It helps us serve you promptly.

If you have no label handy, print OLD address here.

Please Print

Name

Company

Address

City State Zip code

Affix Label **Print NEW address here**

Name

Company

Address

City State Zip code

MAIL COMPLETED CARD TO:

INTERFACE™

P. O. Box 1234
Cerritos, California 90701

Experience the excitement of owning the finest *personal* computer – IMSAI 8080

Waiting for you — all the incredible performance and power of the IMSAI 8080. And at a price you would normally pay for a home music system: $699 in kit; $931 assembled.

WORLD OF USES

The IMSAI 8080 is a superbly engineered, quality computer. It is versatile, expandable and powerful, putting literally hundreds of applications and uses at your fingertips. Imagine sitting at your desk and enjoying interaction with your IMSAI 8080! Press the *on* switch and you're ready for game playing, research, education, business applications, or for laboratory instrument control. It has all the power you need, and more, to make your application ideas come alive.

GROWS WITH YOU

The IMSAI 8080 is designed for many years of pleasure. With its open-chassis engineering you can expand your system by adding peripherals and interfaces. The 22-slots and 28 amp power supply mean that you can plug-in today's, plus *tomorrow's* modules.

Right now you can add a module for displaying color graphics and characters on TV; a ready-to-use keyboard; small and large printers, and a single interface that lets you attach multiple devices including a cassette tape recorder. Expect the latest, exciting equipment from IMSAI. We are committed to leadership in this expanding technology.

EASY TO PROGRAM

With our BASIC language you can operate the IMSAI 8080 quickly and easily. Technically knowledgeable? Use our assembly language to develop sophisticated and unique software.

If you're thinking personal computer, treat yourself to the very best – IMSAI 8080

Send for free four-color brochure or $1 for catalog. Call or write for name of nearest dealer.

IMSAI

IMSAI Manufacturing Corporation
14860 Wicks Blvd.
San Leandro, CA 94577
(415) 483-2093
TWX 910-366 7287

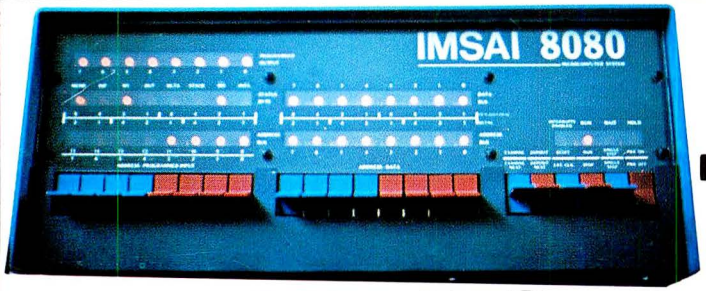
European Distributor
Harper Industry Products, Ltd.
6079 Sprendlingen 2
West Germany

Prices: USA Domestic

ULTRABYTE

PRESENTS

THE FORMULA FOR POWER



+ Z-80

CPU BOARD

= IMSAI Z-80

THAT'S RIGHT! THIS ULTRA-POWERFUL COMBINATION IS OFFERED BY ULTRABYTE, YOUR FACTORY AUTHORIZED DEALER OF ALL THESE QUALITY COMPONENTS

PLEASE RUSH ME THE FOLLOWING:

- ☐ *IMSAI 8080 Standard Kit, (without Standard CPU Board) with Powerful Z-80 CPU Board ..... $ 849.00
- ☐ *Same as above (IMSAI plus Z-80), but with 22 Slot Mother Board ..... $ 914.00
- ☐ IMSAI 8080 Kit, Standard ..... $ 599.00
- ☐ IMSAI 8080 Kit, with 22 Slot Mother Board ..... $ 660.00
- ☐ With all 22 Connectors, add $80.00 ..... $ 740.00
- ☐ Card Guides, 20 pairs ..... $ 20.00
- ☐ IMSAI 8080 Kit, with pre-soldered 22 Slot Mother Board Containing all 22 Connectors ..... $ 820.00
- ☐ IMSAI 8080, Standard Assembled ..... $ 950.00
- ☐ IMSAI 8080, 22 Slot Mother Board, with all 22 Connectors, Completely Assembled and Tested $1193.00
- ☐ 4K Bytes Static Memory, Kit ..... $ 129.00
- ☐ Assembled ..... $ 231.00
- ☐ ICOM's Microfloppy Disk System, including Drive, Power Supply, Cabinet, Controller/Interface Card, Manuals, Diskette & Software ..... $ 895.00
- ☐ ICOM's Frugal Floppy, including Disk Drive, CF 360 Controller, all Cables and Connectors $1129.00
- ☐ ICOM's Frugal Floppy, same as above, but with Dual Disk Drive $1749.00

SUB-TOTAL ..... _____

PRICES SUBJECT TO CHANGE WITHOUT NOTICE

*PHONE IN YOUR CREDIT CARD ORDERS
*CALIF. RES. ADD 6% SALES TAX
*MOST ORDERS SHIPPED WITHIN 30 DAYS
*SHIPPING CHARGE: COLLECT F.O.B. FACTORY

Enclosed Check or Money Order
For $ _____
Bankamericard No. _____
Mastercharge No. _____
Interbank No. ____
Signature: _____

Name: _____

Address: _____

City, State, Zip: _____

UltraByte Incorporated
P.O. Box 370, Westminster, CA. 92683 2683
(714) 892-4679
Executive Offices:
Park Plaza Building, Suite 17281
Plaza Street, Westminister, CA. 92683
(Send Self-addressed, stamped envelope for complete computer accessory listj)

ARTWORK COURTESY OF GO FOR IT GRAPHICS, BOX 532, MANHATTAN BEACH, CA 90266

CIRCLE NO. 12 ON INQUIRY CARD